(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 June 2001 (14.06.2001)

PCT

(10) International Publication Number
WO 01/42949 A2

(51) International Patent Classification[7]: G06F 17/00

(21) International Application Number: PCT/GB00/04180

(22) International Filing Date: 31 October 2000 (31.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/169,620 8 December 1999 (08.12.1999) US

(71) Applicant *(for all designated States except US)*: BROKER-TO-BROKER NETWORKS, INC [US/US]; 1209 Orange Street, City of Washington, County of Newcastle, DE (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: GIESSEN, Charles, Richard [US/GB]; 32 South Eaton Place, London SW1 9JJ (GB).

(74) Agent: MURGITROYD & COMPANY; 373 Scotland Street, Glasgow G5 8QA (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/42949 A2

(54) Title: SYSTEM FOR FACILITATING TRANSACTIONS ON AN EXCHANGE

(57) Abstract: A broker-to-broker system enables originating brokers to send orders for execution to fulfilling brokers using broker-to-broker system secure networks and systems and for the resulting transactions to be cleared and settled in the name of the two parties using broker-to-broker system to manage the process. Locale-specific format conversions, such as for securities numbers, language, and currencies, are automatically performed.

SYSTEM FOR FACILITATING TRANSACTIONS ON AN EXCHANGE

FIELD OF THE INVENTION

The present invention relates to a system which may be used to facilitate transactions on an exchange.

BACKGROUND OF THE INVENTION

There are many types of exchange on which members communicate to transact business. For example, members of financial exchanges may trade money or an equivalent in exchange for an instrument or commodity. Other types of trading exchanges include the many electronic business-to-business exchanges and business-to-consumer exchanges for trading goods and services. Such business-to-business exchanges typically adopt either a vertical or a horizontal structure. Vertical exchanges are based on specific industry sectors, e.g. aerospace, automotive or chemicals. An example of a vertical business-to-business exchange is GMtradexchange.com on which parts for the automotive industry are traded. Horizontal business-to-business exchanges are organized around the products and services provided which typically span more than a single industry sector. An example of a horizontal business-to-business exchange is MRO.COM which provides a market place for materials, repair and operations goods and services.

Exchanges of the above general types are used by members. The term "member" used herein refers to a party having access to an exchange in order to transact business on it. The term "exchange" used herein refers to any type of exchange on which business can be transacted.

Although computer systems embodying the present invention can be used with any type of exchange, the exemplary embodiment described herein relates to securities trading exchanges (including traditional stock exchanges and alternative trading systems (ATSs) and electronic crossing networks (ECNs)). Principals or agents such as stock brokers, investment banks or appropriately regulated asset managers having direct or indirect access to securities exchanges in order to buy and sell on behalf of investors are referred to herein as "brokers". For the purposes of the description the term "broker" is synonymous with the term "Member".

In recent years, the U.S. market for equity securities has experienced dramatic growth. This growth is evidenced by the trading volumes in the

NASDAQ, NYSE and AMEX markets, as well as trading volume in the so-called Third Market:

The average daily volume of securities traded in the NASDAQ increased from 225.0 million shares in December 1992 to 867.1 million shares in December 1999.

The average daily volume of securities traded on the NYSE increased from 222.2 million shares in December 1992 to 692.8 million shares in December 1998.

The average daily volume of securities traded on the AMEX increased from 14.2 million shares in December 1992 to 40.21 million shares in December 1998.

The trading volume and growth in all of these markets are driven by many factors, including increased cash flows into equity-based mutual funds; historic high returns in U.S. equity markets; the emergence and rapid growth of on-line discount brokers; technological innovations, such as the emergence of the Internet, and reduced transaction costs.

While large trading volumes have provided brokers with an opportunity to spread fixed costs over a larger number of trades, net profits per trade, however, have declined. As a result, broker-dealers and other financial institutions are seeking new trading methodologies to identify and take advantage of the profit opportunities represented by each trade. These broker-dealers also seek to increase their order flow. Order flow is the volume of buy and sell orders received by a broker-dealer. Increased order flow provides market makers with a greater number of trades and increased trading profit opportunities. To succeed, these broker-dealers require sophisticated trading methodologies, computerized trading systems, and risk management practices that can handle increased order flow while maintaining low costs per trade. They also need personnel with the requisite expertise to deliver superior trade executions and customer service.

For example, people who need to buy/sell securities for investment purposes, *i.e.*, Investors, issue their orders to buy/sell securities to a financial advisor/intermediary who is authorized to carry out securities dealings, *e.g.* a broker or bank. The broker then fulfils the Investor's order generally in one of two ways:

(a) If the broker has access to an exchange where the securities are transacted, the fulfilling broker goes into that market and executes the order. This action is the Domestic activity. The broker then delivers the proceeds to the investor. This is the "traditional" stock-brokering activity, which has been taking place for the last 70-100 years;

(b) If the broker does not have access to an exchange where the securities are transacted, typically the case for small or medium-sized brokers when asked to obtain foreign securities, the broker communicates with another broker who is a member and that broker executes the order. In this case the first broker is a customer of the second broker. In this case the second broker delivers the proceeds to the first broker, who then in turn delivers the proceeds to the investor. However, logistical barriers currently make cross-border securities transactions difficult, error-prone and expensive.

SUMMARY OF THE INVENTION

Systems embodying the present invention seek to provide improved systems for facilitating transactions.

According to a first aspect of the present invention there is provided a method of operating a computer system to facilitate transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the method comprising:

receiving at a first interface of a processing system at least one information item of an electronic transaction proposal from an originating party;

transmitting at least one information item relating to the electronic transaction proposal from a second interface of the processing system to a fulfilling member;

generating at the processing system settlement criteria to be accepted by the originating party and the fulfilling member; and

receiving from each of the originating party and the fulfilling member an indication of acceptance of the settlement criteria generated by the processing system.

Preferably the step of receiving the at least one information item of the electronic transaction proposal comprises receiving an information item indicating what is to be transacted. The step of receiving the at least one information item of the electronic transaction proposal may also comprise receiving an information item identifying a designated fulfilling member to perform the transaction. The settlement criteria generated by the processing system may be based on settlement information received from the originating party and the fulfilling member. The settlement criteria generated by the processing system may also be based on stored settlement information accessible by the processing system. The settlement information received from each of the originating party and the fulfilling member includes an indication of acceptance of the settlement criteria generated by the processing system. The processing system can generate settlement instructions to be issued on behalf of the originating party and the fulfilling member.

Preferably settlement instructions are generated responsive to said indications of acceptance of the settlement criteria.

In preferred embodiments a fulfilling member can request a modification to at least part of an electronic transaction proposal from an originating party.

The step of receiving the at least one information item of the electronic transaction proposal may comprise receiving information items relating to a proposed transaction selected from one or more of the following:

a transaction type indicator;

a quantity indicator;

a price condition; and

timing information.

The timing information can indicate a proposed transaction date and/or a proposed settlement date.

The step of receiving the at least one information item of the electronic transaction proposal may comprise receiving information items relating to the proposed transaction from the originating member in two or more sub-steps. For example, an originating party may provide a first information item in a first sub-step and the processor system can supply a further relevant information item responsive thereto. The processing system can supply a plurality of further relevant information items from which the originating party can select. In another case, an originating member provides a first information item indicating what is to be transacted in a first sub-step and the processing system prompts the originating party to select from further information items representing exchange options in a further sub-step.

The processing system can convert at least a portion of an electronic transaction proposal from a first format appropriate to the originating party into a second format appropriate to the fulfilling member. For example, the processing system converts a transactional term in a first language appropriate to the originating party into a second language appropriate to the fulfilling member. The processing system may convert a price indication in a first currency appropriate to the originating party into a corresponding price indication in a second currency appropriate to the fulfilling member. If the exchange is a securities exchange the processing system can convert a first security identifier appropriate to the originating party into a second security identifier appropriate to the fulfilling member. The first format is selected by the originating party. The second format is selected by the fulfilling member. The processing system can also transfer an information item between the first interface and the second interface without altering the format of the information item.

Advantageously, preferred embodiments of the processing system can generate a predetermined signal to alert a member of a change in status of a proposed transaction.

In many embodiments, the fulfilling member also functions as an originating party by submitting transaction proposals for execution on a further exchange. Likewise, the originating member also functions as a fulfilling member by performing a transaction on a further exchange.

A computer system for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the system comprising:

a first processing system interface adapted to receive one or more information items of an electronic transaction proposal from an originating party;

a second processing system interface adapted to transmit one or more information items relating to the electronic transaction proposal to a fulfilling member;

a processing system for routing communications including information items to and from the first and second interfaces, the processing system being operable to generate settlement criteria to be accepted by the originating party and the fulfilling member, and wherein the processing system is arranged to receive first and second information items indicating acceptance of the settlement criteria by each of the originating party and the fulfilling member.

According to another aspect of the present invention there is provided a computer system for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the system comprising:

a first processing system interface adapted to receive one or more information items of an electronic transaction proposal from an originating party;

a second processing system interface adapted to transmit one or more information items relating to the electronic transaction proposal to a fulfilling member;

at least one further processing system interface adapted to issue settlement instructions in respect of a transaction performed by the fulfilling member; and

a processing system for routing communications including information items to and from the first and second interfaces, the processing system being operable to generate settlement instructions to be issued via the at least one further interface on behalf of the originating member and the fulfilling member.

According to another aspect of the present invention there is provided a method of operating a computer system to facilitate transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the method comprising:

receiving at a first interface of a processing system an information item of an electronic transaction proposal from an originating party, said information item being in a first form appropriate for the originating party;

converting said information item of the electronic transaction proposal from the first form into a second form appropriate to a fulfilling member; and

transmitting the information item in the second form from a second interface of the processing system to a fulfilling member.

According to another aspect of the present invention there is provided a computer system for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the system comprising:

a first processing system interface adapted to receive an information item of an electronic transaction proposal from an originating party, said information item being in a first form appropriate to the originating party;

a processing system operable to convert the information item of the electronic transaction proposal from the first form into a second form appropriate to a fulfilling member; and

a second processing system interface adapted to transmit the information item in the second form to the fulfilling member.

According to another aspect of the present invention there is provided a computer readable medium having stored therein a set of general purpose routines for facilitating transactions on exchanges, the computer readable medium comprising:

a first routine for receiving at least one information item of a transaction proposal from an originating party;

a second routine for transmitting at least one information item of the transaction proposal to a fulfilling member;

a third routine for generating first and second settlement criteria to be accepted by said originating party and said fulfilling member; and

a further routine for receiving first and second indications of acceptance of said settlement criteria from said originating party and said fulfilling member.

According to another aspect of the present invention there is provided computer program code for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to an exchange on behalf of an originating party not having access to the exchange, the program code comprising:

a first set of instructions for receiving at least one information item of a transaction proposal from an originating party;

a second set of instructions for transmitting at least one information item of the transaction proposal to a fulfilling member;

a third set of instructions for generating settlement criteria to be agreed by each of said originating party and said fulfilling member and for receiving first and second indications of agreement from said originating party and said fulfilling member.

According to another aspect of the present invention, there is provided a method of facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the method comprising:

receiving at least one information item of a transaction proposal from an originating party;

transmitting at least one information item of the transaction proposal to a fulfilling member;

generating settlement criteria to be accepted by the originating party and the fulfilling member; and

receiving indications of acceptance of the settlement criteria from each of said originating party and said fulfilling member.

Preferably, the method includes generating first and second settlement instructions on behalf of said originating party and said fulfilling member responsive to receipt of said indications of acceptance.

According to another aspect of the present invention there is provided a system for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the system comprising:

means for receiving a transaction proposal from an originating party;

means for transmitting a transaction proposal to a fulfilling member;

means for receiving accepted settlement particulars from said originating party and said fulfilling member;

means for generating first and second settlement instructions on behalf of said originating member and said fulfilling member responsive to receipt of said accepted settlement particulars; and

means for issuing said first and second settlement instructions.

According to another aspect of the present invention there is provided a method of facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the method comprising:

receiving an information item of a transaction proposal from an originating party, said information item being in a first form appropriate for the originating party;

converting said information item of the transaction proposal from the first form into a second form appropriate to a fulfilling member; and

transmitting the information item in the second form to a fulfilling member.

According to another aspect of the present invention there is provided a system for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange; the method comprising:

means for receiving an information item of a transaction proposal from an originating party, said information item being in a first form appropriate to the originating party;

means for converting said information item of the transaction proposal from the first form into a second form appropriate to a fulfilling member; and

means for transmitting the information item in the second form to a fulfilling member.

According to another aspect of the present invention there is provided computer program code for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to an exchange on behalf of an originating party not having access to the exchange, the program code comprising:

a first set of instructions for receiving an information item of a transaction proposal from an originating party, said information item being in a first form appropriate for the originating party;

a second set of instructions for converting said information item of the transaction proposal from the first form into a second form appropriate to a fulfilling member; and

a third set of instructions for transferring the information item in the second form to a fulfilling member.

According to another aspect of the present invention there is provided a computer readable medium having stored therein a set of general purpose routines for facilitating transactions on exchanges, the computer readable medium comprising:

a first routine for receiving an information item of a transaction proposal from an originating party, said information item being in a first form appropriate for the originating party;

a second routine for converting said information item of the transaction proposal from the first form into a second form appropriate to a fulfilling member; and

a third routine for transmitting the information item in the second form to a fulfilling member.

Preferred embodiments provide a system, method, and software for broker-to-broker trading, in which an order for a proposed transaction is received from an originating member, indicating a designated fulfilling broker. The order is in a format appropriate to the originating broker. The order is converted into in a format appropriate to the fulfilling broker and transmitted to the fulfilling broker and settlement instructions are issued from a central location. As a result, preferred embodiments deal with local-specific conversion and settlement issues are automatically handled, thereby increasing the efficiency of cross-border transactions, reducing transaction costs, fostering increased transparency, and increasing the volume and liquidity of international securities transactions. Preferred embodiments thus provide a high value, robust messaging and transaction processing infrastructure that will enable executing-brokers ("Fulfilling Members" (FMs)) to service orders directed to them and provide brokers placing orders ("Originating Members"(OMs)) variable lower-cost access to securities markets and transaction services worldwide.

Still other objects and advantages of preferred embodiments of the present invention will become readily apparent from the following detailed description, simply by way of illustration of the best modes contemplated of carrying out the invention. As will be realized, the invention is capable of other and different embodiments, and its several details are capable of modifications in various obvious respects, all without departing from the invention. Accordingly, the drawings and description are to be regarded as illustrative in nature, and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIGURE 1 is a schematic diagram illustrating a broker-to-broker communication system embodying the present invention;

FIGURE 2 is a schematic diagram illustrating key information flows in a system according to Figure 1;

FIGURE 3 is a schematic diagram of an end terminal for use in the broker-to-broker system of Figure 1;

FIGURE 4 is a schematic diagram of a computer system implementing the broker-to-broker system of Figure 1;

FIGURE 5 is a flow chart illustrating connection and display update processes of the computer system of Figure 4;

FIGURE 6 is a flow chart illustrating order placement processes of the computer system of Figure 4;

FIGURE 7 is a flow chart illustrating order acceptance/rejection processes of the computer system of Figure 4;

FIGURE 8 is a flow chart illustrating execution reporting processes of the computer system of Figure 4;

FIGURE 9 is a flow chart illustrating settlement initiation and processing procedures of the computer system of Figure 4;

FIGURE 10 is a schematic diagram illustrating another example of a computer system suitable for implementing the broker-to-broker communication system of Figure 1;

FIGURE 11 is a schematic diagram of an end terminal and local network for use in the computer system of Figure 10;

FIGURE 12 schematically illustrates a first interface architecture of the computer system of Figure 10;

FIGURE 13 schematically illustrates a second interface architecture of the computer system of Figure 10;

DESCRIPTION OF THE PREFERRED EMBODIMENT

A broker-to-broker trading system and methodology are described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

DEFINITION OF TERMS

The financial industry is beset with terminology that varies with jurisdiction. Thus, in order to prevent confusion and because the broker-to-broker system is an entirely novel arrangement within the financial industry, i.e. it is not a bank, brokerage, exchange, advisor, or any other kind of conventional financial related entity, it is vital that clarity of purpose, role and responsibility of the components, systems and entities within the broker-to-broker system be maintained at all times.

Originating Party/Member The originating party/member is the entity on the broker-to-broker system's network that originates orders into the network. The originating party may be any institution appropriately regulated to place orders with a fulfilling member on behalf of themselves or third parties. The originating member is called thus because this entity can be a member of an exchange who starts the process off by entering an order into the system, i.e. it originates the order. It is assumed that typically orders entered into the broker-to-broker system network have been entered because the originating member has been requested by a customer to purchase/sell securities. Strictly speaking however, there is no requirement for the originating member to enter only their customer's requests, in addition some members may conduct extensive "own-account"/proprietary trading using the broker-to-broker system's network to send orders for execution. In examples in this document the originating party/member is designated as Party A.

Fulfilling Member The fulfilling member is the entity on the broker-to-broker system's network which receives orders from the originating member and then executes against those orders, i.e. the order is the desire to do something and fulfillment is the satisfying of that desire. In most securities markets, the fulfilling member has the following options available to them as to how to fulfil the order or parts thereof.

- On Exchange, i.e. by buying/selling securities from/to another exchange member. The fulfilling member may or may not expose the order to the entire market depending on the rules of the market.

- Off Inventory, i.e. by buying/selling securities from/to the fulfilling member own inventory of securities.
- Agency Cross, i.e. when the fulfilling member determines that there are two or more orders on opposite sides of each other, (i.e. one order is for a sale and another order is for a purchase of the same set of securities), the member matches the two orders.
- Third party crossing network, i.e. when the fulfilling broker sends the order into a third party crossing network such as Instinet, and Posit.

In examples in this document the fulfilling member is designated as Party B.

Member Members are customers (users) of the broker-to-broker system having access to one exchange or another and are usually brokers acting as transacting agents for investors. When referring to the clients of the broker-to-broker system's member's the term Investor will be used implying the eventual beneficiary of the proceeds of transactions. Generally, investors have no contractual or any other kind of relationship with the broker-to-broker system; an investor's relationship legal or otherwise exists between the investor and their financial advisor(s).

Investor The term used for entities that have contracted with one of the broker-to-broker system's members to carry out services. Investors do not usually contract with the broker-to-broker system, nor would they directly use the services of the broker-to-broker system - see "Product" below for how investors may use technology created by the broker-to-broker system, in order to access the broker-to-broker system's services/products even indirectly, the investor has a relationship with the broker-to-broker system customer who is responsible or liable for all the actions of the investor.

Street The term used to denote entities within the financial community that will interact with the broker-to-broker system's members typically to provide them with research and execution counterparties

Service The term used to denote ongoing activities carried out by the broker-to-broker system personnel, and/or its agents and partners for which the broker-to-broker system's member's will pay some kind of fee.

Product The term used to denote an item(s) which may be sold to, licensed to, leased to, or rented to the broker-to-broker member which that member may then use to offer a service to his/her clients, e.g. investors. Optionally the broker-to-broker system through one of its facilities management operating subsidiaries and in conjunction with its partners and agents may be selected by the member to install, configure and operate the product on behalf of and in the name of that member.

Order An order is an instruction from one party to another party to obtain/dispose of securities on behalf of the issuing party in return for some payment. Typically unless an order is fulfilled, no transaction will be deemed to have taken place.

Proceeds Proceeds are the cash monies or securities which parties to a transaction will deliver/receive at the conclusion of the transaction.

Settlement Settlement is the process of delivering the proceeds of a transaction into the custody of the beneficiaries of the transaction, e.g. cash into a bank account, securities into a securities account.

Execution Whenever a securities transaction takes place, an execution is deemed to have taken place, i.e. one party agrees to sell securities to another party in exchange for remuneration. The terms "fill" or "completion" are also often used instead of execution particularly when many small securities transactions are required to fulfil an order.

Custodian Securities are not always held by investors in the form of paper certificates. Most developed markets insist that securities are held in de-materialized form in a depository, (often a computer system managed by some quasi-government institution or association of banks, which keeps track of who is the beneficial owner of the security). In such cases, the depository may nominate, approve and supervise certain commercial organizations to provide the interface between the depository and the owners of securities - these organizations are the custodians who maintain custody of the securities on behalf of the actual owners.

(Custodians may also maintain vaults where paper certificates are stored).

SYSTEM OVERVIEW

FIG. 1 represents a schematic overview of a preferred communications system. The broker-to-broker computer system is an order routing and settlement facilitation system, which comprises an integrated network of desktop applications through which an originating member 204 seeking fulfillment of an investor 210's securities transaction order may enter the order for delivery to a designated fulfilling member 208 who is qualified to fulfill the order or part thereof on the basis of prevailing conditions in the fulfilling members' market 212, subject to the conditions specified by the originating member 204. In this embodiment, a broker-to-broker network arrangement comprises two secure interfaces OMIF 202 and FMIF 206 connecting originating members 204 and fulfilling members 208, respectively, to a broker-to-broker processing system 200 (having "order management" capabilities), which processing system acts as a transaction manager tracking the status of orders. Additional integrated applications generate settlement instructions, collect and report status information from other parts of the broker-to-broker system, provide access for the correction of problems of the broker-to-broker system, and perform data storage and retrieval functions. Members are able to use the broker-to-broker system as originating members 204 or fulfilling members 208, or both.

It will be appreciated that many originating members 204 and fulfilling members 208 can be connected to the broker-to-broker processing system 200 and that the interfaces 202 and 206 can be any type necessary to achieve this.

Using the interfaces 202 and 206 connecting them to the broker-to-broker system, originating members 204 may enter information concerning a proposed transaction. If all required information is properly provided, the order is then sent to an order management system for routing and delivery to the fulfilling member 208. As this embodiment is a broker-to-broker system designed to enhance the ability of domestic originating members 204 to send appropriate orders to fulfilling members 208 qualified to execute such trades on foreign markets, the processing System 200 is able to perform certain data conversion or translation services of a clerical or ministerial nature when a cross-border or inter-market trade calls for such services.

The combined order routing and settlement management features of the broker-to-broker processing system 200 afford a seamless and efficient inter-exchange order management system. Conversion service features of the processing system 200 provide opportunities for increased efficiency of cross-border and inter-market transactions and thus contribute to broader access to securities markets worldwide, in particular because the processing system 200 offers those capabilities at lower variable costs to members who have previously been less able to fully take advantage of advanced technologies to overcome the costs and other burdens arising from the numerous additional variables associated with international securities transactions. It is particularly advantageous that settlement instructions are issued from a single source. Accordingly, the use of the broker-to-broker network by members may further increase the transparency of international securities markets and afford to investors through their brokers increased liquidity and efficiency in conducting securities transactions.

All communications are routed through the broker-to-broker processing system 200. All communications sent to and from the broker-to-broker system may be encrypted using 128 bit SSL technology to ensure complete confidentiality. Once an order is received, a fulfilling member 208 designated by the originating member 204 will review the information for completeness and accuracy, as will be described in more detail hereinafter. The fulfilling member 208 can then determine in its sole discretion whether to accept or reject the order. Once an order is accepted, the fulfilling member 208 will then seek to execute the order on the specified market satisfying the terms of the order. The fulfilling member and the originating member communicate through the broker-to-broker system or through any other means they may choose as to the status of fulfillment of the order and any modifications thereto. When the members agree that the order has been fulfilled (or the unfulfilled portion cancelled), information concerning the transaction is sent to the originating member 204 for review. The agreed transaction can then be submitted by both members to the settlement management portion of the broker-to-broker processing system 200 which generates settlement instructions, based entirely on transaction details and settlement account (and preference) information previously provided by members, for transmission to settlement agents designated exclusively by the respective members.

Additional functions to be performed by the broker-to-broker processing system 200 include the generation of execution reports, and similar functions as may be desired by brokers using the broker-to-broker computer system.

KEY SYSTEM FLOWS

FIG. 2 schematically depicts the system flows for the broker-to-broker system, showing how an order moves through the various network elements until the proceeds are delivered to one of the broker-to-broker system's members.

The investor issues an order to his/her financial originating member who enters the order into an originating member interface OMIF 202. Referring to Fig. 2, the order 10 is then transmitted through the processing system 200 via an order management sub-system 16 to a fulfilling member via a fulfilling member interface FMIF 206. The fulfilling member executes the order in his/her market and notifies the originating member of the results by sending a fill message 12 to the originating member via the originating member interface OMIF 202. When both members agree that the order has been fulfilled, a message 14 detailing the transaction is sent to a settlement management system 20 which completes the process by issuing settlement instructions 22 according to the SWIFT protocol, or another suitable protocol, thereby ensuring that both parties deliver/receive the securities, and that monies are paid by the members to/from each other.

At all stages in the transaction cycle, a customer service system 24, is notified by all of the other network elements of the processing system 200 of the status of the transaction so that inquiries may be easily and rapidly dealt with.

HARDWARE OVERVIEW: END TERMINAL

FIG. 3 is a block diagram illustrating a computer system 400/402 which is used as an end terminal of both the originating member and the fulfilling member in the exemplary embodiment described herein. Computer system 400/402 includes a bus 102 or other communication mechanism for communicating information, and a processor 104 coupled with bus 102 for processing information. Computer system 400/402 also includes a main memory 106, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 102 for storing information and instructions to be executed by processor 104. Main memory 106 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 104. Computer system 400/402 further includes a read only memory (ROM) 108 or other static storage device coupled to bus 102 for storing static information and instructions for processor 104. A storage device 110, such as a magnetic disk or optical disk, is provided and coupled to bus 102 for storing information and instructions.

Computer system 400/402 may be coupled via bus 102 to a display 112, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 114, including alphanumeric and other keys, is coupled to bus 102 for communicating information and command selections to processor 104. Another type of user input device is cursor control 116, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 104 and for controlling cursor movement on display 112. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allow the device to specify positions in a plane.

In this embodiment communication with the broker-to-broker processing system is facilitated by browser software running on the computer system 400/402 by means of the processor 104 executing one or more sequences of one or more instructions contained in the main memory 106. Such instructions may be read into main memory 106 from another computer-readable medium, such as storage device 110. Execution of the sequences of instructions contained in main memory 106 causes processor 104 to perform the process steps described herein. One or more processors in a multi-processing arrangement may also be employed to execute the sequences of instructions contained in main memory 106.

The term "computer-readable medium" as used in this context refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media include, for example, optical or magnetic disks, such as storage device 110. Volatile media include dynamic memory, such as main memory 106. Transmission media include, for example, coaxial cables, copper wire, fiber optics and radio interfaces. Transmission media can thus also take the form of acoustic or light waves, such as those generated during radio frequency (RF) and infrared (IR) data communications. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, DVD, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 104 for execution. For example, the instructions may initially be borne on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a communication link such as telecommunications link with or without using a modem. A modem local to computer system 400/402 can receive the data on a communication link and use an infrared transmitter to convert the data to an infrared signal. An infrared detector coupled to bus 102 can receive the data carried in the infrared signal and place the data on bus 102. Bus 102 carries the data to main memory 106, from which processor 104 retrieves and executes the instructions. The instructions received by main memory 106 may optionally be stored on storage device 110 either before or after execution by processor 104.

Computer system 400/402 also includes a communication interface 118 coupled to bus 102. For example, communication interface 118 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a telecommunications line. Communication interface 118 provides a two-way data communication coupling to a communication line 120 comprising part of the originating member interface OMIF 202 or the fulfilling member interface FMIF 206, depending on whether the computer 400/402 is an end terminal belonging to an originating member or a fulfilling member. Fixed or leased telecommunications lines are used to connect the end terminals 400/402 to the broker-to-broker processing system 200. The minimum fixed-line speed available for the broker-to-broker system is at present 64Kbits/second. Higher speeds are installed in proportion to the number of concurrent broker-to-broker system sessions in use at the member's location, and the response times which they wish to achieve. As a rule of thumb, 2 sessions can be concurrently handled per 64Kbit/sec segment.

The interfaces OMIF 202 and FMIF 206 thus connect originating and fulfilling members, respectively, to the broker-to-broker processing system 200 which acts as an order placing/accepting mechanism, a transaction manager for tracking the status of orders, and a means of automatically issuing settlement instructions. Additional integrated applications collect and report status information from other parts of the system, provide access for maintenance and the correction of system problems, and perform data storage and retrieval functions. End terminals 400 of the originating and fulfilling member interfaces OMIF 202 comprise a web technologies based front-end (or graphical user interface) which the members interact with and which connects to an application core for processing the data entered by the respective member. Preferred embodiments also include a computer-to-computer applications programmer interface OMIF-API so the originating member can interface his/her computer systems directly into the broker-to-broker system.

FMIF 206, the system which interfaces the broker-to-broker system to the fulfilling broker 208, provides him/her with the ability to interact with the broker-to-broker system and its other customers. Like OMIF 202, FMIF 206 comprises: a web technologies based front-end, through which the fulfilling member interacts with the broker-to-broker system and an application core. Preferred embodiments also include a computer-to-computer applications programmer interface, FMIF-API, which the fulfilling member may use to interface his/her systems directly into the broker-to-broker system network so as to eliminate re-keying of data.

In this embodiment, the front-end software (GUI) uses the following standards: Netscape Navigator 4.5, Microsoft Internet Explorer 4; HTML 4: JavaScript; JAVA; 128 Bit SSL. The application core may be built using the following standards: Netscape Commerce or Apache Server; CGI with Perl and C/C++ 504; Java servelets; Java servelet pages; application server (e.g. WebLogic); SUN Solaris or Linux Operating System; Oracle Database;. Tellurian Secure Sockets.

The target technical environment at the originating/fulfilling member end terminal will thus be Netscape Navigator 4.5 or Microsoft Internet Explorer 4.01, higher versions may be used. Cookies, small files stored on the user's local hard-drive, are commonly used by web based applications to store data about the user and their preferences so that when the user next logs on, the system can skip the steps required to obtain that data for the new session. The use of cookies within the OMIF 202 and FMIF 206 interfaces can be severely restricted if not eliminated so as to minimize any security risk. That is, items like colors, and language preference, frame sizes and positions can be safely stored in cookies, but items such as security identifiers, default fulfilling members and default execution venue are preferably not be stored since knowledge of these could be used to interpret an originating member's transaction history.

The OMIF 202 and FMIF 206 are constructed so as to require only minimal resources at the end terminals 400,402. That is, they have an operating "footprint" on the user's machine which is as lightweight or small as possible. It must not be considered the norm that users will be technically literate and that they will have high powered machines. A typical end terminal might be a Pentium personal computer running the Microsoft Windows NT operating system.

HARDWARE OVERVIEW: BROKER-TO-BROKER PROCESSING SYSTEM

Figure 4 illustrates the preferred architecture for the broker-to-broker processing system 200. Figure 4 also shows how the end terminals of an originating member 400 and a fulfilling member 402 are connected to the broker-to-broker processing system 200 by a secure network 404. The secure network 404 comprises at least the communications lines 120 from the end terminals 400 and 404.

The broker-to-broker processing system 200 comprises a web server 410, an application server 420 provided with a front end database 425, an order management system (OMS) server 430 provided with an OMS database 435, and a settlement management system (SMS) server 440 provided with a SMS database 445. The OMS database 435 and the SMS database 445 are connected by a database link 452. The web server 410 is connected to a messaging server 480 and the application server 420 is connected to a security server 470. The interface between the web server 410 and the application server 420 operates based on the Java RMI protocol. The order management system server 430 is connected to the application server 420 via an interface defined according to the Orbix CORBA protocol.

The web server 410 includes a submission processor 412 and a page generator application 414. The web server 410 supports a range of internet-based server technologies including, for example, hyper text mark-up language (HTML), JavaScript, extensible mark-up language (XML), and other protocols for the definition of page content, format and functionality. It should be appreciated that the web server is not connected to the internet in this embodiment.

The application server 420 includes a processor (not shown), order and query data access software 422, order manager software 424 and order storage software 423. The application server 420 operates hosting and data access services for the web server 410 to create pages which are implemented as Enterprise Java Beans. The front end database 425 connected to the application server 420 holds records of orders 426, executions 427, requests 428 and user control information 429. The application server 420 controls page generation through the web server 410 using the order manager software 420.

The order management system server 430 includes a request processor 432, order publisher software 434 and an order processor 436. The OMS database 435 holds records of orders 437, executions 438 and requests 439. The order management system server 430 can transfer data to the front end database 425 via the order publishing software 434 and the order storage software 423. Accordingly, data held in the OMS database 435 can be replicated into the front end database 425 following an access to the OMS database 435 by the order processor 436. The order management system server 430 can also access the SMS database 445 via the database link 452.

The FIX server 490 is provided in order to implement application programmer interfaces, as will be explained in more detail hereinafter.

The settlement management system server 440 has a trade processor 442 capable of generating settlement instructions in the form of, for example, SWIFT format messages 443 and fax messages 444. The SMS database 445 holds records of settled trades 446, stock information 447, market information 448, member information 449 and commission rate information 450.

The settlement management system server 440 polls the SMS database 445 at predetermined time intervals to identify transactions for which settlement instructions can be generated. The SMS database 445 receives an intra-day price feed every half-past the hour, where the price given is the price of the hour (e.g. the price at 10.00am is delivered at 10.30am). The days closing prices are also fed in nightly.

As will be explained hereinafter, the settlement management system server 440 need only become involved when the settlement particulars have been agreed by the parties to the transaction. Although the various software and hardware components of the broker-to-broker processing system 200 are illustrated as separate functional components for reasons of clarity, they may of course be provided in a different configurations.

The application server 420 and the front end database together define an "application subsystem" of the computer system on which the core application software runs. The core application software provides content to the end terminals. The order management system server 430 and the OMS database 435 together define an order management sub-system which, as the name suggests, is responsible for validating each originating member order; keeping track of the status of orders; controlling, tracking, and applying conversions/translations to the order; entering the business rules associated with translations; and dispatching completed orders for settlement processing. Similarly, the settlement management system server 440 and the SMS database 445 can be regarded as comprising a settlement management sub-system. The first action taken by the order management sub-system is to apply stringent/rigorous content validation to each order. The order management sub-system can access information in the settlement management sub-system, as may be required, and provide messages and other content to the end terminals via the application server 420, front end database 425 and the web server 410.

Once an order passes the content validation checks of the order management sub-system, it is routed to the fulfilling member specified by the originating member. All interactions with orders are passed via the order management sub-system which ensures that predefined rules of business are not violated. For example, the order management sub-system will prohibit members from entering an executed quantity greater than the order quantity; it will prohibit multiple members from applying changes to the order simultaneously; it will ensure that the state rules of an order cannot be violated.

The order management sub-system facilitates and controls the following actions performed by either the originating member (OM) and/or the fulfilling member (FM) in respect of an order in the state specified:

State of Order:	**OM**	**FM**
New Order Entered	Cancel	Accept; Reject
Accepted Order	Cancel; Modify	Fill; Cancel; Modify
OM (cancel) Request	Withdraw	Accept; Reject
FM (cancel) Request	Accept; Reject	Withdraw
OM Modification Request	Withdraw	Accept; Reject
FM Modification Request	Accept; Reject	Withdraw
OM Sign-off Request	Withdraw	Accept; Reject
FM Sign-off Request	Accept; Reject	Withdraw
FM (cancel) Fill Request	Accept; Reject	Withdraw
Cancelled	-	-
Rejected	-	-
Signed off	-	-

The order management sub-system also routes any requests against an order from the originating party to the receiving party; and conversely the accept/rejection of a request is routed back to the originating party, as will be explained further hereinafter.

OPERATION

Because the front end software may be used at many sites, and very often where the level of technical support is minimal or self-help is the norm, it preferably does not require the installation of any components from disk/CDROM etc at the customer's premises. As will be explained, the front end software accessed via end terminals 400 of the OMIF 202 may include the following frames or capabilities: Order entry; order manager (indicating status of orders, cancel, modify, sign-off); intra broker-to-broker system member messaging (e-mail, interactive text messaging, e.g. ICQ, IRC, interactive voice, video messaging, e.g. net meeting); transaction history report generation; lookup facilities for securities information; look-up facilities to locate other members on the broker-to-broker system; and an interface to the broker-to-broker system on-line electronic library.

The front end software accessed via the end terminals 402 of the FMIF 406 includes the following frames or capabilities: order acceptance; execution entry; transaction manager (status of orders, cancel, modify, cancel fill, sign off); intra broker-to-broker system customer messaging; (e-mail; interactive text messaging, e.g. ICQ, IRC; interactive voice, video messaging, e.g. net meeting;) transaction history report generation; look-up facilities for securities information; Look-up facilities to locate other members on the broker-to-broker system; interface to the broker-to-broker system on-line electronic library.

As the system is designed in part to enhance the ability of domestic originating members to send appropriate orders to fulfilling members qualified to execute such trades on foreign markets, the system can perform certain data conversion or translation services of a clerical or ministerial nature when a cross-border or inter-market trade calls for such services. Accessed through drop-down menus viewed on the GUI of the front end, the conversion service options available to originating members include: retrieval from data storage and inclusion in the message of appropriate additional securities identification numbers (e.g., the system can provide the applicable ISIN number when given a particular CUSIP number for securities identified for trading in multiple jurisdictions); the reformatting of orders to comply with the customs, dictates or market vernacular of a particular exchange to be used by the designated fulfilling member (e.g., "market held" from a U.S. broker would be converted to "best," the equivalent expression used by broker-dealers in Great Britain); language translation (e.g., English to French) of essential transaction terms (e.g., "buy" or "sell"); and the generation and inclusion by the system of additional available information requested by an originating member that is not an element of a proposed transaction, but offers convenience to the parties, such as display of indicative-only currency conversion data. The system also offers members an option to include free form messages which are not subject to any modification, translation or conversion by the system.

Figure 5 is a flow chart illustrating how both originating members and fulfilling members can view relevant information (information specific to the member) on their respective end terminals 400, 402. The end terminals 400, 402, are in this embodiment personal computers located at a site owned by the member. As outlined above, the end terminals 400, 402, are provided with a browser application capable of: displaying pages of order information; generating messages; and generating reports. Originating and fulfilling members establish a connection and log-on to the broker-to-broker processing system 200 in the same way. For security purposes every user of the system use a physical security token unique to each individual at the originating member's premises, such as SecurID from RSA Dynamics, to identify themselves to the system. This token and operating guide may be delivered to the member using secure courier facilities. Once in possession of the token, all that may be required of the user may be to start up their web browser and navigate to the broker-to-broker URL . If there are software components to be installed the broker-to-broker system will automatically detect that condition and carry out the necessary operations remotely.

Referring to step 500, the member first establishes a connection to the web server 410 of the broker-to-broker processing system 200 from an end terminal 400/402. The member then enters authentication details including a user name, personal identification number PIN, and secure identity number.

At step 502, the authentication details are passed from the web server 410 to the security server 470 via the application server 420. The security server 470 can then process the authentication details (see step 504). If the member is not authorized to use the system or has provided incorrect authentication details, a "log-in error" message is generated and sent to the member's end terminal (step 506).

If the member is an authorized user and has input his/her authentication details correctly, the security server accepts the login and the application server 420 references the user control information in the front end database 425 to establish whether the member is an originating member or a fulfilling member and what information is to be displayed on the screen of the member's end terminal (see step 508). At step 510, the order manager software 424 of the application server 420 provides the information necessary for the page generator 414 of the web server 410 to display screen information specific to the member at the end terminal 402. This information is then sent to the end terminal for display to the member.

The information content of the member's screen is refreshed at predetermined time intervals by the application server 420 referencing the front end database 425 and re-transmitting updated information to the end terminal responsive to a request by the end terminal 400/402. The system is configured such that end terminals request a screen content update at 20 second intervals. However, it will be apparent that other time intervals can be used (see step 512). It will be apparent that at any given point in time each member sees a screen display having content defined by the relevant records in the front end database 425 as they existed at the time of the last refresh operation. The content itself is defined by the current order status, while the format may be at least partly defined by member preferences.

The order entry frame viewable by originating members enables originating members to enter an order(s) for transmission to a fulfilling member for execution. Items on the order entry frame may include: originating member's investor reference code; transaction type - buy/sell indicator; security identifier; market where security is to be executed; quantity of securities to be executed; price conditions which will govern the execution; timing applicable to the order; duration applicable to the order; identity of the fulfilling member; designation of the execution venue; payment/receive currency; free form message entry area (viewable by both originating member or fulfilling member); intended trade date; intended settlement date; free form text entry area (viewable by originating member only). Further frames allow originating members to enter: identity of the securities account into/from which securities will be delivered/removed; and identity of the cash account into/from which monies will be delivered/debited;

Connected members see an orders list on the left hand side of an order management screen. Clicking on an order in the list brings up further details of the order and enables predefined actions to be taken. An originating member can use his end terminal 400, for example, to:

- enter orders for transmission to the selected fulfilling members;
- request to cancel his/her orders;
- request to modify attributes of his/her orders;
- respond to requests to cancel initiated by the fulfilling member;
- respond to requests to modify initiated by the fulfilling member;
- request to sign off his/her orders i.e. request to initiate the settlement process for the fills/executions against his/her orders;
- respond to requests to signoff his/her orders initiated by the fulfilling member;
- receive fill/execution advisories/reports;
- review the status of his/her orders;
- provide the information/data required by broker-to-broker system to carry out settlement management on their behalf;
- upload material, e.g. notes, pricing models/tolls/guidelines, audio clips, video clips, to broker-to-broker system central electronic information library;
- search the electronic information library and download material;
- interrogate their billing account with broker-to-broker system to obtain information

such as: transaction fees outstanding to broker-to-broker system, customers' membership fees outstanding, rebates applicable, fee structure agreed with broker-to-broker system, average trader per day (TPD) for week, month, quarter;

- send messages directly to other broker-to-broker system customer(s) using the secure, guaranteed network facilities;
- send messages directly to the customer support system;
- search the data repository for securities information;
- search the data repository for member information;
- obtain information about their use of broker-to-broker system's facilities such as: transaction history (by: time period, security, market, exchange, sector, geographical region), execution quality statistics, volume weighted average price (VWAP), *i.e.* the price taking into account the size of the transactions executed), delta from best bid or best offer;
- security control (authorized names, passwords).

Figure 6 is a flow chart illustrating how an order may be placed by a member functioning as an originating member.

At step 600 the originating member OM enters, on his/her terminal 400, a security identifier (SID) to identify the security he/she wishes to buy or sell. The OM clicks the "order" button displayed at his/her terminal. The market field is a view-only field, i.e. the user cannot type into this field. An originating member's graphic user interface will only allow securities to be entered in those markets where the broker-to-broker system has signed up fulfilling members as customers since these are clearly the only markets where the order can be transmitted to. A security identifier, SID, along with the market reference code, uniquely identifies the security which the originating member wants to transact. The choice of numbering agency for the SID can be the choice of the originating member. It cannot be sufficiently stressed how easy (ergonomic/user-friendly) determining/looking-up SIDs is using the front end software. At a minimum, the broker-to-broker system will support the following SIDs /numbering agencies: CUSIP (Corp. of United States Instrument Code); ISIN (International Standard for Instrument Numbering); LOCAL (Local Exchanges Identifier); SEDOL (Stock Exchange Daily Order List -- London Stock Exchange Instrument Code); or full company name of the security.

At step 604, a message indicating which security is to be bought or sold is sent from the originating member to the order management system server 430 via the web server 410 and the application server 420. The order management server 430 then accesses the settlement management system database 445 via the OMS database 435 and the database link 452 (see step 606). The results of the access are passed from the order management server 430 to the application server 420.

In many cases, securities are listed on multiple exchanges in multiple jurisdictions. The security listed is not a different security, i.e. possession of that security wherever purchased confers the same rights on the holder. Securities such as American Depository Receipts (ADRs) , and Global Depository Receipts (GDRs) are different securities from the underlying and while they may be converted back to the original are still for the purposes of reporting etc, different, distinct securities in their own right. An example of multiple listing is Reuters stock which is listed on other exchanges along with its primary listing on the London Stock Exchange. In these cases the security normally has the same ISIN.

If the application server 420 cannot uniquely identify the security to be traded based on the results of the database query, a message is sent from the application server 420 to the end terminal 400 containing information on the choices of markets for trading the securities. The end terminal 400 displays a choice of markets for the security intended to be traded by the originating member on the screen of the end terminal 400 (see step 608). At step 610, the originating member selects from the choice of markets for trading the security by clicking the screen.

Thus, if a SID is entered which causes a multiple match of securities held within the SMS database 445, the full company name of the matching securities along with their traded market of execution will be displayed to the originating member. The originating member then chooses the market he/she wishes to enter the security order on. This will set the market. The requirement for the market to be set is that the broker-to-broker system needs to know to which market the order is being transmitted so that the correct validation checks may be applied, and so that the appropriate conventions and language translations may take place. The order process can then continue as it would have done had the application server been able to uniquely identify the security at step 606 (see below).

If the application server 420 can uniquely identify the security selected by the originating member, the originating member is prompted on screen to input particulars of his/her order (see step 612). At step 614, the originating member defines the order by entering specific details of the desired order. Originating members enter information concerning the proposed transaction. In addition, to a valid security identifier, the information includes: the number of securities; whether the order is for a purchase or sale - a buy/sell indicator; valid price conditions for the specific market and security; valid duration and timing of the order; a suggested trade date for the transaction; a suggested settlement date for the transaction; a valid designated fulfilling member; a desired execution venue; the originating member account details for settlement; and valid authenticated identity of the issuing person at the originating member.

The originating member may need to keep track of orders entered so that he/she can relate them back to an order(s) which he/she may have been given by a customer. To this end an investor reference code is entered against an order. (Even when the order is for their own-account, i.e. proprietary trading, they may need to track the order). For small originating members, this investor reference code may very well be the same as a customer account. However larger operations may have computer systems.

The originating member may enter an alphanumeric string which is their reference code for the order. By convention the client reference code should be used as the reference code. The broker-to-broker system need not in any way interpret/alter this client reference code. The presence or otherwise of this reference code shall have no bearing on how the broker-to-broker system processes the order or its resulting fills. Note also that while this reference code will be electronically attached to the order and all resulting transactions relating to the order, only the originating member will ever view the reference code. This code is not viewable by the fulfilling member. The broker-to-broker system Customer Support personnel, will view the reference code as a field of asterisks, i.e. *****************. Even within the broker-to-broker systems it is recommended that this field be rendered non-easily readable to prevent accidental disclosure to the broker-to-broker system technical personnel. This approach helps to ensure that the confidentiality of the investor's details is maintained.

The parties involved in the transaction may need to unequivocally identify an order. To this end, the system will generate a unique order reference number which is viewable by all parties, including the customer support desk. This reference code field enables the originating member to put code(s) into his/her own systems so that he/she can reconcile the order and its resulting executions with his/her own systems. This further reference code can be quoted to all parties to the order to unequivocally identify the order in question. For example when the originating member and the fulfilling member are discussing the order, or either party request help from the Customer Support desk, this reference code may allow all parties to rapidly identify the order in question leading to quicker resolution of the problem. This reference code may be visible in human readable form to all parties on the broker-to-broker system.

The quantity item allows the user to specify the number of securities to be bought or sold. Short forms to speed up entry and minimize errors are provided on the OMIF, e.g. T for thousands, M for millions, H for hundreds. In addition where lot sizing and/or minimum order sizing applies, broker-to-broker system front end will caution the user whenever the lot size entered is unlikely to be acceptable by the intended market. (The front end software can display the lot sizes applicable).

The Buy/Sell indicator specifies the intended direction of the transaction, i.e. whether the order is for a security is to be bought or sold. The indicator represents the direction from the perspective of the originating member, i.e. if the originating member has an order to obtain securities for his/her customer, then the indicator will show BUY. Conversely if the originating member has an order to dispose of securities on behalf of his/her customer, then the buy/sell indicator will show SELL. The price conditions item, allows originating members to specify the price conditions which he/she is prepared to accept when the fulfilling member executes the transaction. The classic price conditions are: sell limit price - i.e. the lowest price that the originating member is prepared to accept in exchange for disposing of his/her securities; buy limit price - i.e. the highest price that the originating member is prepared to pay in exchange for obtaining the securities; sell at market - i.e. the originating member is prepared to receive whatever is the highest current prevailing price in the market; buy at market - i.e. the originating member is prepared to pay whatever is the lowest current prevailing price in the market.

Other items comprised in the order entry frame as viewed from an originating member's end terminal 400 include: timing applicable to the order; duration applicable to the order; identity of the designated fulfilling member; the commission rate which the fulfilling member will charge to the OM for executing the order; designation of the execution venue; preformatted instructions to the fulfilling member indicating how the order should be traded; free format message area to the FM; free format message area viewable by the OM only; trade date; settlement date; capacity the OM is acting in (agency/principal); payment/receive currency.

The system enables the originating member to select among the universe of fulfilling members available for a given security or market. The system need not select a fulfilling member on behalf of the originating member, or change the identity of the originating member's selected fulfilling member on an order. Where an originating member selects a fulfilling member that does not have the capability to trade a particular security or on the particular market in the originating member's order, the originating member chooses a fulfilling member appropriate for that security or market before the order can be further processed.

After entering the order details the originating member is prompted to confirm they are correct (see step 614 of Figure 6). If all required information is properly provided, the order is then sent via the broker-to-broker system for verification and for delivery to the fulfilling member.

At step 616, a message containing the details of the originating member's order is sent to the order management system server 430. The order processing software 436 of the order management system server 430 then stores the details of the order in the OMS database 435 (see step 618). The order management system server 430 can transfer the order information stored in the OMS database 435 to the front end database 425 via the order publishing software 434 and the order storage software 423 of the application server 420. At step 620, details of the originating member's order are stored in the front end database 525 connected to the application server 420.

Operable through the interface's 202,206, for example, using dropdown menus, the conversion service options available to originating members 204 will include: retrieval from data storage and inclusion in the message of appropriate additional securities identification numbers (e.g., the broker-to-broker processing system 200 will be able to provide the applicable ISIN number when given a particular CUSIP number for securities identified for trading in multiple jurisdictions); the reformatting of orders to comply with the customs, dictates or market vernacular of a particular exchange to be used by the designated fulfilling member (e.g. "market held" from a U.S. broker would be converted to "best," the equivalent expression used by broker-dealers in Great Britain); language translation (e.g., English to French) of essential transaction terms (e.g., "buy" or "sell") and the generation and inclusion by the processing system of additional information requested by an originating member 204 that is not an element of a proposed transaction, but offers convenience to the parties, such as display of indicative-only currency conversion data. The processing system 200 also offers members an option to include messages which are not be subject to any modification ("free form" messages).

Thus, each originating member, and each authorized person acting on behalf of a member, can choose and set the numbering agency which they wish to use to identify securities on their orders. The broker-to-broker system will translate the entered SID into the SID required by the fulfilling broker (and vice versa) and other parties to the transaction. Internally within the broker-to-broker system, securities will be identified by the broker-to-broker system financial instrument identifier (FID). The broker-to-broker system FID is a unique code generated by the broker-to-broker system's data repositories which takes into account for example multiple listings of the same security, (one of the reasons the broker-to-broker system cannot use ISIN internally). The SID entry field allows the originating member to look-up the SID code using the security's text name, or part thereof, e.g. if the user enters "British" in the SID field, an instruction is sent to the order management server to search the SMS database security database and display a list of securities whose full name starts with British. The user then selects the required security. As a convenience, the front end software will remember at least the last 100 SIDs submitted by an originating member, and present that list, together with the text name of the SIDs, to the user on a drop-down list for selection.

Using an end terminal to access front end software the fulfilling member can:

- receive and accept orders from originating members for fulfillment;
- request to cancel his/her orders;
- request to modify attributes of his/her orders;
- respond to requests to cancel initiated by the originating member;
- respond to requests to modify initiated by the originating member;
- request to sign off his/her orders (i.e. request to initiate the settlement process);
- respond to requests to sign off his/her orders initiated by the originating member;
- transmit fill/execution advisories/reports to the originating member;
- request to cancel a fill/execution advisory;
- withdraw any outstanding request which they have initiated;
- review the status of his/her orders to be worked;
- provide the information/data required by the broker-to-broker system to carry out settlement management on their behalf;
- upload material, e.g. notes, pricing models, tools, guidelines, audio clips, video clips to the broker-to-broker system central electronic information library;

- search the electronic information library and download material;
- interrogate their billing account with the broker-to-broker system to obtain information such as: transaction fees outstanding to the broker-to-broker system, customer fees outstanding, rebates applicable, fee structure agreed with the broker-to-broker network administrator, average traders per day TPD for week, month, quarter;
- send messages directly to other members using the secure, guaranteed the broker-to-broker system network facilities;
- send messages directly to the customer support system;
- search the data repository for securities information;
- search the data repository for member information;
- obtain information about their use of the broker-to-broker system's facilities such as: transaction history (by: time period, security, market, exchange, sector, geographical region);
- effect security control (authorized names, passwords).

Figure 7 illustrates how a fulfilling member becomes aware of orders placed with him and may accept or reject them. A fulfilling member may become aware of new orders placed with him when he establishes a connection to the broker-to-broker system 200 or when the screen content of the end terminal 402 is refreshed in a period after new order particulars have been transferred to the front end database 425.

Referring to step 700, the fulfilling member sees all new orders along with all pending orders on an orders list on the left-hand side of his screen. The fulfilling member can select one and view the order particulars in detail.

If the fulfilling member elects to view the particulars of a new order placed with him, the fulfilling member will click on the order within the order list whereby the application server 420 references the front end database 425 and the order manager software 424 supplies details of the order to the end terminal 402 via the page generator 414 in the web server 410. Fields presented on this screen are read-only except where otherwise noted. The order acceptance frame enables the fulfilling member to accept an order(s) for fulfillment. Items on the order acceptance frame may include: means for the fulfilling member to accept or reject the order; originating member's order reference code; transaction type - buy/sell indicator; security name; security identifier; market where security is to be executed; quantity of securities to be executed; price conditions which will govern the execution; designation of the execution venue; payment/receive currency; trade date; settlement date; and commission rate; free form message area. Other frames allow the fulfilling member to provide the identity of the securities account into/from which securities will be delivered/removed; and identity of the cash account into/from which the monies will be delivered/debited;

When an order is transmitted to the fulfilling member, an alert mechanism, visual and audible, prompts the fulfilling member that there is an order to be dealt with. Failure to respond to the alert within a pre-determined time-period may cause the order to be automatically cancelled and the originating member so notified. This time-period for acknowledgement is configurable on the following basis: system wide; per market; per fulfilling member.

Electronic acknowledgement of the order allows the broker-to-broker system 200 to send a positive indication back to the originating member that the order has been delivered to the designated fulfilling member and is being considered for acceptance. This order acknowledgement does NOT mean that the fulfilling member has agreed to fulfil the order, i.e. no contract has yet been formed between the originating member and the fulfilling member. Once an order has been acknowledged, details of the order are placed onto the FMIF pending orders manager screen. The order having been acknowledged, all parties now "know" that the order has been seen by the fulfilling member and could be fulfilled. The fulfilling member now has to accept, reject, or request modification to the order within a configurable time-period. This acceptance time-period is configurable on the following basis: system wide; per market; per fulfilling broker.

The Buy/Sell indicator specifies the intended direction of the transaction to the fulfilling member. The Security Identifier, SID, along with the market reference code, uniquely identifies the security which the fulfilling member is to transact on behalf of the originating member. Automatic reformatting and conversion/translation functions of the broker-to-broker function may mean that the content of certain fields viewed by the fulfilling member is different to that input by the originating member. The format and content type of the subject matter viewed at an end terminal is selected/configured by the member who uses the end terminal. For example in the case of a fulfilling member, the choice of numbering agency for the SID is primarily the choice of the fulfilling member. The broker-to-broker processing system will map the originating member's SID into the form required by the fulfilling member. At a minimum, the broker-to-broker system will support the following SIDs /numbering agencies: CUSIP, ISIN, SEDOL, LOCAL (Local Exchange Identifier) and full company name. Each fulfilling member, and each authorized person at the fulfilling member's premises, can choose and set the numbering agency which they wish to use to identify securities on the orders which they receive. The broker-to-broker system will, if required, convert the originating member's SID into the SID required by the fulfilling member and other parties to the transaction. Internally, i.e. within the broker-to-broker system, securities will be identified by the broker-to-broker system FID. The broker-to-broker system FID is a unique code generated by the broker-to-broker system 's data repository which takes into account for example multiple listings of the same security.

A security to be transacted by a fulfilling member is uniquely identified by means of the security identifier SID and the market. The quantity item specifies the number of securities to be bought or sold by the fulfilling member. Where lot sizing and/or minimum order sizing applies, the FMIF front end software expresses the quantity as an absolute quantity and as a number of lots to sell/buy. The price conditions item indicates the price conditions the originating member is prepared to accept when the fulfilling member executes the transaction. Examples of price conditions are: sell limit price - i.e. the lowest price that the originating member is prepared to accept in exchange for disposing of his/her securities; buy limit price - i.e. the highest price that the originating member is prepared to pay in exchange for obtaining the securities; sell at market - i.e. the originating member is prepared to receive whatever is the highest current prevailing price in the market; buy at market - i.e. the originating member is prepared to pay whatever is the lowest current prevailing price in the market.

Other items accessible to the fulfilling broker include: identity of the fulfilling member; the commission rate which the firm will charge to the originating member for executing the order;

designation of the execution venue; preformatted instructions from the originating member indicating how the order should be traded; free format message area from the originating member; identity of the originating member who routed the order; trade date; settlement date; capacity the originating member is acting in; payment/receive currency; transaction management; intra member messaging; interface to the broker-to-broker system electronic library.

The fulfilling member reviews the information for completeness and accuracy. If the need arises he can contact the originating member via a communication means of the broker-to-broker system. The fulfilling member can then determine in its sole discretion whether to accept or reject the order. The fulfilling member is presented with on-screen buttons inviting him/her to accept or reject the order (see step 702).

When a fulfilling member rejects an order, the originating member is sent an order reject message together with an indication of the reason for the rejection. That is, the fulfilling member is prompted to indicate a reason for rejecting the order and an order reject message including the reason is sent from the end terminal 402 to the order management system server 430 via the web server 410 and the application server 420 (see step 704). At step 706, the order record in the OMS database 435 is accessed to add an indication that the order has been rejected. An indication that the order has been rejected is then replicated to the front end database

425 via the order publisher software 434 and the order storage software 423 of the order management system server 430 and the application server 420, respectively (see step 708). The status of the order within the broker-to-broker system is thus changed to "rejected". Step 710 illustrates that the originating member's screen content is changed to indicate "order rejected" instead of "order placed" next time the screen content is updated.

When a fulfilling member accepts an order, a contract is now deemed to exist between the originating member and the fulfilling member. The contract is governed by the terms of the order, as specified in the fields of the order, the terms of the fulfilling members agreements with the broker-to-broker system, and is subject to the regulations of the fulfilling member's jurisdiction, e.g. the member agrees to the desired quantity, price conditions, delivery conditions and execution venue specified by the originating member. Thus, if the fulfilling member accepts an order, an order acceptance message is sent from the end terminal 402 to the order management system server 430 (see step 712). The order processor 436 of the order management system server 430 then updates the relevant order record in the OMS database 435 to indicate acceptance of the order by the fulfilling member (see step 714). An indication that the order has been accepted is then transferred from the OMS database 435 to the front end database 425 by means of the order publishing software 434 and the order storage software 423 (see step 716). Next time the screen content of the originating member (or the fulfilling member) terminal is updated by one of the periodic references to the front end database 425, the display of the end terminal 400 (or 402) will indicate an "order pending fill" status to indicate acceptance (718).

Once the originating member is notified of acceptance it can be assumed that the fulfilling member will seek to execute the order according to the terms specified. No item on the order may now be altered without requesting the fulfilling member to positively accept the changes requested.

In the case that the order is accepted, the fulfilling member seeks to execute the order on the specified market satisfying the terms of the order. The fulfilling member and the originating member can communicate through the broker-to-broker system or through any other means they may choose as to the status of fulfillment of the order and any modifications thereto. When the members agree that the order has been fulfilled or the unfulfilled portion cancelled, information concerning the transaction is sent to the originating member for review. The agreed transaction can then be submitted by both members to the settlement management portion of the system which then generates settlement instructions, based entirely on transaction details, settlement account and preference information previously provided by members, for transmission to the settlement agents designated exclusively by the respective members.

Figure 8 is a flow chart illustrating how the originating member is made aware that the fulfilling member has executed an order placed with him. Referring to the step designated 800, the fulfilling member selects the order he has executed from the list displayed on the screen of end terminal 402. To provide the fulfilling member with more details of the order he has selected, the application server 420 references the front end database 425 and content relevant to the order selected is supplied to the end terminal 402 via the web server 410 (see step 802). The fulfilling member can then enter execution data (see step 804). At step 806, the execution data supplied by the fulfilling member are sent to the order management system server 430 and are recorded by the order processor 436 in the OMS database 435 (see step 808). The execution data are subsequently transferred to the relevant record in the front end database 425 via the order publishing software 434 and the order storage software 423 (see step 810). The "executed" status of the order is then indicated on the displays of both the originating member and the fulfilling member end terminals next time the screen content is refreshed.

If the order has been fully executed by the fulfilling member, either the originating member or the fulfilling member may initiate a settlement procedure (see step 810). If on the other hand the order has been partially executed by the fulfilling member, the fulfilling member can execute the remainder later. If the fulfilling member executes the remainder of the order later (see step 812), the process from step 800 is followed to report completion of order to the originating member. In cases where the fulfilling member does not execute the remainder of the order that day, the order status is preserved until the next suitable day.

Figure 9 is a flow chart illustrating how settlement instructions are generated and issued by the broker-to-broker computer system. Either party can initiate the settlement procedure, however in this example the procedure is initiated by the fulfilling member. Referring to step 900, the fulfilling member selects the executed order for which he wishes to initiate settlement proceedings by clicking on it. The fulfilling member's screen prompts him/her to "request sign-off" (see step 902). When a fulfilling member selects the request sign-off button, he/she is prompted to enter settlement information, including data identifying the settlement account of the fulfilling member (see step 904). At step 906, the settlement information provided by the fulfilling member is sent from the end terminal 402 to the order management system server 430 from where it is provided to the settlement management system server 440. The order management system server 430 transfers the fulfilling member's settlement information to the SMS database 445 via the OMS database 435 and the database link 452. The OMS server 430 makes a database stored procedure call into the SMS database 445 passing in the details of the relevant trade, responsive to which the OMS server 430 is provided with the calculated commissions, and any local market charges, taxes or levies (see step 908). At step 910 the calculated settlement criteria are sent directly to the fulfilling member terminal 402 where the fulfilling member can review them and either accept or reject them.

If the fulfilling member accepts the settlement criteria, a message containing the accepted settlement particulars is sent to the order management system server 430 which generates a message changing the status at the order management system server 430. Certain particulars, e.g. average prices and security settlement account information is stored in the OMS database 435. These settlement particulars are transferred into the front end database 425 via the order publisher software 434 of the OMS server 430 and the order storage software 423 of the application server 420. The status of the order is thus updated to be "sign off requested" which will be displayed to the originating member and fulfilling member the next time screen content on the terminals 400, 402 is refreshed (912).

Referring to step 914, the originating member clicks on the "sign-off requested" button and is prompted to accept/reject the request. The OM can accept and provide settlement information, including information identifying the originating member's settlement account (see step 915). After the originating member has entered his/her settlement information, the settlement information is sent to the order management system server 430 for processing based, for example, on for example average price information provided by the fulfilling member (see step 916). At step 918, the order management system server accesses the SMS database 445 by means of a stored procedure call passing the trade details into the SMS database 445. In response, calculated settlement criteria relating to the originating member are supplied from the SMS database to the OMS database 435 via the database link 452 (see step 918). The order management system server 430 transfers the calculated settlement criteria of the originating member to the end terminal 400 where the originating member can review and either accept or reject them (see step 920).

If the settlement criteria are accepted by the originating member, a message containing the accepted particulars is sent from the end terminal 400 back to the order management system server 430 and the accepted particulars are stored in the OMS database 435 (see step 922). The front end database 425 is updated by means of the order publisher software 434 and the order storage software 423 in the usual way, after which update the screen content can reflect the new status of "signed off".

Once both the originating member and the fulfilling member have accepted the settlement criteria of a given trade, the order management server creates two trade records (see step 924). One trade record corresponds to a buy transaction and is associated with the appropriate one of the originating or fulfilling members. The other trade record corresponds to a sell transaction and is associated with the other one of the originating or fulfilling members. The two trade records are stored in the SMS database 445 via the OMS database 435 and the database link 452.

The settlement management server 440 polls the SMS database 445 at predetermined intervals to find trade records which require processing. Having detected the trade records the SMS server 440 validates certain order attributes (e.g. security, market, market listing, trade date, price, settlement date, broker identities, commission rates and default charges) against static data held in the SMS database 445. The initiation of the settlement procedures is undertaken by the generation of agent instructions during the trade verification process. Referring to step 926, the trade processor 442 of the settlement management server 440 processes the settlement management information of that transaction to generate settlement instructions (see step 926).

In this way, generic trade instructions are generated and associated with a given medium that identifies how the message will be sent to the clearing agent. The settlement management sub-system thus derives the company's stock and cash settlement agent and the client's stock and cash settlement in order to ascertain whether the trade will be settled either against or free of payment. Considerable flexibility is available for defining settlement preferences for the company and its counterparties.

Settlement instructions can be established and varied for a wide range of criteria such as particular markets, instrument types, individual stocks, and currencies. A first settlement instruction message is sent to the settlement agent designated by the originating member and a second settlement instruction message is sent to the settlement agent designated by the fulfilling member. The settlement instructions may be transmitted to settlement agents in, for example, SWIFT format messages 443 or by fax message 444. (SWIFT is an international standard recognized by banks for sending messages to make payments of cash and securities.) It will be apparent that criteria and criteria acceptance messages may also be routed between the originating member, the fulfilling member via indirect routes. For example, the fulfilling member's indication of acceptance may instead be sent directly to the originating member along with criteria for acceptance by him/her. Both indications of acceptance can then be returned to the processing system together. In the context, of receiving and sending information items the terms "to originating party/fulfilling member" and "from originating party/fulfilling member" should be construed accordingly.

Thus, appropriate settlement instructions are generated automatically by the settlement management part 440,445 of the broker-to-broker processing system 200 for all parties to the trade and sent to the relevant settlement agents. Settlement instructions relating to agreed settlement particulars are issued from a single source and at an appropriate point in time. In this preferred embodiment, settlement instructions are issued to the settlement agent of the originating member and the settlement agent of the fulfilling member at the same time.

In preferred embodiments the originating member interface OMIF 202 includes an application program interface, referred to herein as BIAPI, designed to allow the broker-to-broker members to interface their systems directly into broker-to-broker system 200 eliminating and minimizing the need to key data into order entry screens, and keying data about fills, positions etc into their systems.

There are two forms of BIAPI: BIAPI-F - File based interface; and BIAPI-I - Interactive interface.

The file based API, BIAPI-F, is designed to enable the widest, simplest and quickest system interconnect possibilities, whilst the interactive API, BIAPI-I, is designed to enable hi-performance, functionally rich system interconnects to be created.

Technically, BIAPI-F consists of text files containing comma separated, quoted tag and value records terminated by an end-of-line character. These files may be read or put from/into the file-space of the terminal 400 client, e.g.

"SNA =CUSIP", (Security Numbering Agency)
"SID =0123456A", (Security Identifier)
"QTY =500", (Quantity)
"PXTYPE =LIMIT", (Pricing Type)
"LIMIT =10.5", (Limit Order Level)

"PAYCUR =EUR" (Payment Currency)

BIAPI-I may be implemented as a component, which exposes methods that are then invoked by the host program.

The functionality provided through either BIAPI-F, or BIAPI-I, will include: send orders for execution; receive fill/execution notification; receive order status; receive settled position(s) and balance(s) information; receive open positions(s) and balance(s) information; receive their account information, i.e. electronic billing from the broker-to-broker system; send settlement account(s) information to the broker-to-broker system; receive transaction history information.

Through BIAPI-I, the originating members are able to carry out all of the functions available through BIAPI-F, and the following additional set of functionality: session control; log on/off; manage transactions; request cancellation; request modification; request sign-off; accept/reject cancellation requests; accept/reject modification requests; accept/reject sign-off requests; withdraw an initiated request; query the interface for order status; security lookup; request for quote; request reports/data with filter specifications; transaction history; open position(s) and balance(s) information; settled positions(s) and balance(s) information; their account information, i.e. electronic billing from the broker-to-broker system.

The API of the FMIF 206, is designed to allow the broker-to-broker system's customers to interface their systems directly into the broker-to-broker system eliminating and minimizing the need to key in order data and keying data about fills, positions etc. into the fulfilling member's trading systems.

There are two forms of API for the FMIF 206 SELAPI: SELAPI-F - File based interface; and SELAPI-I - Interactive interface.

The file based API, SELAPI-F, is designed to enable the widest, simplest and quickest system interconnect possibilities, whilst the interactive API, SELAPI-I, is designed to enable hi-performance, functionally rich system interconnects to be created.

SELAPI-F comprises text files containing comma separated, quoted tag and value records terminated by an end-of-line character. These files may be read or put from/into the file-space of the client terminal 402, e.g. "SNA =CUSIP"'"SID =0123456A","QTY =500","PXTYPE =LIMIT","LIMIT =10.5","PAYCUR =EUR" etc.

SELAPI-I comprises a component, which exposes methods that are then invoked by the host program.

The functionality provided through either SELAPI-F, or SELAPI-I, will include: receive orders for execution; transmit fill/execution notification; receive settled position(s) and balance(s) information; receive open positions(s) and balance(s) information; receive their account information, i.e. electronic billing from the broker-to-broker system; send settlement account(s) information to the broker-to-broker system; receive transaction history information.

Through SELAPI-I, the fulfilling members are able to carry out all of the functions available through SELAPI-F, and the following additional set of functionality: session control; log on/off; manage transactions; request cancellation; request modification; request sign off; accept/reject cancel requests; accept/reject modification requests; accept/rejection sign off requests; withdraw an initiated request; request cancellation of fills; query the interface for order status; security lookup; respond to quote requests; request reports/data with filter specifications; transaction history; open position(s) and balance(s) information; settled positions(s) and balance(s) information; and their account information, i.e. electronic billing from the broker-to-broker system.

TECHNOLOGY STANDARDS

The broker-to-broker network and interfaces adhere to as many standards as possible for the following reasons: lower overall cost due to economics of mass-market production and sales; standards based technologies tend to have fewer problems/bugs, i.e. more dependable; easier to resource personnel; leverage of existing technology intellectual property; ability to stay current with and leverage technological changes; and ease of interfacing to other systems both internally and externally to the broker-to-broker network.

In this embodiment, the standards chosen by the broker-to-broker system are as follows:

Netscape Navigator, Microsoft Internet Explorer - for Web browsers

HTML 4, DHTML, SSL, JavaScript - for Web front-end programming

Netscape, Apache - for Web servers

BEA Systems WebLogic - for application servers

C/C++, Java, Perl - for application processing engines

UNIX - SOLARIS, HP/UX, LINUX - for application processing engines

Windows NT - for in-house desktop technology

ORACLE RDBMS - for databases

SUN Microsystems, COMPAQ - for application hardware

COMPAQ - for in-house desktop hardware

STRATUS - for non-stop 100% dependability hardware

TCP/IP - for base level networking protocols

SWIFT - for payment instruction protocol

FIX - for order management protocol

Microsoft Office - for document creation

Adobe Acrobat - for document distribution

RSA Dynamics SecurID - for authentication controls

The broker-to-broker system transaction rules, legalities and processes.

Of course, the functionality embodying the present invention can be achieved with any suitable combination of network elements and protocols.

In this embodiment the broker-to-broker processing system 200 includes the following characteristics:

The processing system 200 does not change the sequencing of orders received from the originating member 204. There is no prioritizing of orders from any given originating member 204, nor any prioritizing of orders among originating member 204. The processing system does not generally change the sequencing of fulfillment advisories transmitted from a fulfilling member 208 back to an originating member 204.

The processing system 200 enables the originating member 204 to select from a plurality of fulfilling members 208 available for a given security or market. Thus, the processing system 200 does not in this example select a fulfilling member 208 on behalf of the originating member 204, or change the identity of the originating member's 204 selected fulfilling member 208 against an order. Where an originating member 208 selects a fulfilling member 204 that does not have the capability to trade a particular security or on the particular market in the originating member's 204 order, the originating member 204 may be required to choose a fulfilling member 208 appropriate for that security or market before the order can be further processed by the processing system 200.

Originating member's 204 may be able to direct and monitor the nature of the fulfillment of their orders through various-mechanisms, initially including at least the following:

The originating member 204 may be able to designate and set as a default the execution venue which the fulfilling member 208 may be required to use when fulfilling the order, for example, on an exchange, off-exchange or a crossing system, for a particular security or member.

In more developed electronic markets, every executed trade reported by the fulfilling member 208 will be time stamped and marked with the best official bid and offer positioned in that market, to enable originating members 204 to compare the execution quality received against market pricing.

The processing system 200 may be able to generate reports of transactions conducted using the broker-to-broker system to help brokers comply with their regulatory reporting obligations.

Settlement of a trade between the originating member and its investor, or between the fulfilling member and the street, can be completely within the province of the member. The system need not generate settlement instructions, confirmations or other forms of advisories or messages to either member's investors or counter-parties.

All transactions that are initiated by communications made through the system may be executed by the members to the transaction in a market and in a manner in which they are qualified to conduct such business. The system is a facility through which information relating to securities orders may be transmitted.

The conversion service features of the system may provide opportunities for increased efficiency of cross-border and inter-market transactions and thus contribute to broader access to securities markets worldwide, in particular because the system may offer those capabilities at lower variable costs to members who have previously been less able to fully take advantage of advanced technologies to overcome the costs and other burdens arising from the numerous additional variables associated with international securities transactions. Accordingly, the use of the system by members may further increase the transparency of international securities markets and afford to investors through their brokers increased liquidity and efficiency in conducting securities transactions.

A flat annual fee may be charged to originating members and, only after a certain number of transactions, an excess usage fee may be charged to originating and fulfilling members whose use of the system results in the generation of a high volume of settlement instructions.

The excess usage fee is not dependent upon the completion of any securities transaction. The settlement service is cost-effective and more error-free given the symmetry of data entered onto the system by the brokers conducting a transaction. The excess usage fee is intended to fairly compensate the broker-to-broker network operator based on a proportional accounting for substantial usage of the system indicated by a greater than anticipated volume of transactions submitted for settlement. Accordingly, unless a system failure or error occurs, no rebate may be provided for an excess usage fee if a transaction submitted for settlement does not in fact close. Indeed, the network operator may reserve the right to charge an additional excess usage fee for further settlement processing when a failure to settle is due to member error. The network operator's receipt of excess usage fees when a certain number of settlement instructions are generated would not cause the network operator to be effecting transactions in securities because the network operator has no substantive role in whether any trade is executed and submitted for settlement, and once submitted, whether the trade successfully closes.

The payment structure contemplated by the network operator reflects the intent for the system to be rigorously neutral as to the effectuation of transactions and to allocate among users their respective share of the costs of operating the system at the service levels demanded in a commercially rational manner. The fees charged by the operator may be based on the operating overhead and the expenses of maintaining the system, plus recapture of start-up costs and a market driven profit.

While this invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

EXAMPLE OF ALTERNATIVE EMBODIMENT

Figure 10 illustrates an example of a further computer system embodying the present invention. BINET 1010 is a secure private network using Internet technology even though it itself need not be implemented on the Internet. Provided appropriate security measures are implemented, such as use of the public key infrastructure (PKI), BINET may comprise the internet in some embodiments, as will be explained below. Thus, this broker-to-broker system can leverage the massive strides taking place in web and communications technology at low cost. Of note is that BINET 1010 interfaces into the core of the broker-to-broker system using the FIX protocol, providing the broker-to-broker system with the ability to replace/upgrade BINET 1010 without unduly impacting the main the broker-to-broker system processing core, and also providing a standard interface into which the broker-to-broker system's larger customers can directly interface their systems.

Order Management System (OMS) 1000 may be a UNIX application, written in C++, an Oracle RDBMS, and Microsoft Windows NT GUI front-end which acts as the transaction manager keeping track of orders, their status and directing the sequence of processing

SELNET 1012 is a secure private network using Internet technology even though it itself need not necessarily be implemented on the Internet. Where appropriate security measures are put in place (e.g. PKI), SELNET may comprise the internet. In this way the broker-to-broker system can leverage the massive strides taking place in web and communications technology at low cost.

SMS, Settlement Management System 1006, may be a UNIX application, written in C/C++, an Oracle RDBMS, and Microsoft Windows NT GUI front-end, which controls the settlement of the proceeds of the transactions. Of key importance is SMS's 1006 ability to automatically generate SWIFT settlement instructions 1080 for all parties to the trade. (SWIFT is an international standard recognized by banks for sending messages to make payments of cash and securities.)

Settlement and Custody 1008 may be external agents (to the broker-to-broker system) who actually carry out the physical act of delivering/receiving cash and securities

CSS, Customer Service System 1004, may be a UNIX and NT suite of applications which collects and reports status from other the broker-to-broker system systems, tracks the relationships with the broker-to-broker system's customers and provides access into the broker-to-broker system's systems to enable service personnel to correct problems.

CDR 1002, Central Data Repository, may be implemented with an ORACLE RDBMS operating under HP/UX hosted on 100% non-stop STRATUS Continuum computer hardware. The CDR 1002 is in this embodiment the single source from which all of the broker-to-broker system's sources static data, and into which all of the broker-to-broker system's systems send status messages. It will be apparent that the key information flows of the computer system of Fig. 10 are similar to those described with reference to Figure 2 and much of the same advantageous functions can be achieved.

Figure 11 illustrates how an end terminal in the computer system of Figure 10 is connected to the broker-to-broker system. The end terminal shown in Figure 10 is generally similar to that in Figure 3 and like reference numerals designate like features.

In this modification, the communication interface 118 is a local area network (LAN) card to provide a data communication connection to a compatible LAN. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software. In any such implementation, communication interface 118 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 120 typically provides data communication through one or more networks to other data devices. For example, network link 120 may provide a connection through local network 122 to a host computer 124 or to data equipment operated by an Internet Service Provider (ISP) 126. ISP 126 in turn provides data communication services through the worldwide packet data communication network, now commonly referred to as the "Internet" 128. Local network 122 and Internet 128 both use electrical, electromagnetic or optical signals that carry digital data streams in accordance with well established protocols such as the TCP/IP protocol suite. The signals through the various networks and the signals on network link 120 and through communication interface 118, which carry the digital data to and from computer system 400/402, are exemplary forms of carrier waves transporting the information.

Computer system 400/402 can send messages and receive data, including program code, through the communications link 120, and communication interface 118. In this Internet example, a server 130 might transmit a requested code for an application program through Internet 128, ISP 126, local network 122 and communication interface 118. In accordance with this embodiment of the invention, such downloaded instruction code provides for implementing a broker-to-broker system as described herein.

The received code may be executed by processor 104 as it is received, for example by using a browser application installed upon the end terminal, and/or stored in storage device 110, or other non-volatile storage for later execution. In this manner, computer system 400/402 may obtain application code in the form of a carrier wave.

It will be apparent that any suitable end terminal device access network arrangement can be used to facilitate communication between a member and the broker-to-broker processing system 200. For example, alternative end terminals include mobile computers and personal digital assistants equipped for operation according to established communication protocols for internet based technologies, such as Wireless Application Environment WAE protocols and later equivalents. Wireless links may be implemented on wireless communication systems based on techniques such as time division multiple access (TDMA), frequency division multiple access(FDMA), space division multiple access(SDMA), code divisional multiple access (CDMA) and hybrids thereof. Some such systems are established and widely used, examples being AMPS and GSM, other such systems are under deployment, such as the Moble Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System (IS95) and the Universal Mobile Telecommunications System (UMTS).

Figures 12 and 13 illustrate alternative access networks BINET and SELNET which may be used by originating and fulfilling members, respectively, to connect to the embodiment of Figure 10.

Referring to FIG. 12, the originating member interface OMIF 202 may replaced with an access network BINET connecting the originating member with the broker-to-broker processing system 200 and providing him/her with the ability to interact with other customers. The access network BINET comprises: an end terminal 1230 through which the originating member interacts with broker-to-broker processing system 200; a computer-to-computer applications programmer interface, BINET-API, which the originating member may use to interface his/her systems directly into the broker-to-broker system network so as to eliminate rekeying of data. BINET-API can be made available, for example, as an interactive interface, BIAPI-I, or as a file/batch interface, BIAPI-F; a secure, encrypted, hi-speed, reliable and guaranteed QOS TCP/IP network connection, BICON, which links BINET sessions at the originating member location into the broker-to-broker system.

Referring to Figure 13, the fulfilling member interface may be replaced by an access network SELNET connecting fulfilling members to the broker-to-broker processing system. Similarly, a computer-to-computer application program interface, SELNET-API can be used by the fulfilling member to interface his/her systems directly into the broker-to-broker system 200. SELNET-API is available as an interactive interface, or as a file/batch interface; a secure, encrypted, hi-speed, reliable and guaranteed QOS TCP/IP network connection, which links SELNET sessions at the fulfilling member's location into the broker-to-broker system.

BIWEB/SELWEB operate on BINET and SELNET, respectively, and comprise two major parts, the front-end or GUI which the fulfilling members interact with, and the application core which processes the data entered by the members.

The front-end may be built using the following standards: Netscape Navigator 4.5., Microsoft Internet Explorer 4.01; HTML 4 + JavaScript; JAVA; 128 Bit SSL.

The application core may be built using the following standards: Netscape Commerce or Apache Server; CGI with Perl and C/C++; Java servelets; Java Servelet Pages; Application Server; SUN Solaris or Linux Operating System; Oracle Database; Tellurian Secure Sockets.

A target technical environment at the fulfilling member may be Netscape Navigator 4.5 or Microsoft Internet Explorer 4.01, higher versions may be used.

Cookies, small files stored on the user's local hard-drive, are commonly used by web based applications to store data about the user and their preferences so that when the user next logs on, the system can skip the steps required to obtain that data for the new session. The use of cookies within the broker-to-broker system can be severely restricted if not eliminated so as to minimize any security risk, e.g. items like colors, and language preference, frame sizes and positions could most probably be safely stored in cookies, but items such as security identifiers, default fulfilling members and default execution venue must not be stored since knowledge of these could be used to interpret the fulfilling brokers transaction history.

BIWEB and SELWEB may be constructed so that its operating "footprint" on the user's machine is as lightweight or small as possible. It must not be considered the norm that users will be technically literate and that they will have high-powered machines. One target machine environment may be a Pentium 233 MHz personal computer running the Microsoft Windows NT operating system.

Because BIWEB and SELWEB will be used at many sites, and very often where the level of technical support is minimal or self-help is the norm, BIWEB and SELWEB must not require the installation of any components from disk/CDROM etc. at the customer's premises. For security purposes every fulfilling member will need to use a physical security token unique to each individual at the fulfilling member's premises, such as SecurID from RSA Dynamics, to identify themselves to the system. This token(s) and operating guide will be delivered to the fulfilling member using secure courier facilities. Once in possession of the token, all that will be required of the user will be to start up their web browser and navigate to the BIWEB or SELWEB URL. If there are software components to be installed BIWEB or SELWEB will automatically detect that condition and carry out the necessary operations.

The access network (referred to herein as BICON) is the physical infrastructure that connects the originating members 1230 with the broker-to-broker processing system 200. In the above embodiment, BICON comprises secure digital communications links between individual members and the broker-to-broker processing system 200. In one modification, BICON is a Virtual Private Digital Network, VPDN which is a physically secure, and information secure, guaranteed quality service, QOS, data communications network. Such networks can be installed between the originating member's location and the nearest broker-to-broker point-of-presence. The contract for installation and operation is between the originating member and broker-to-broker systems. In another modification, BICON comprises the internet.

Access to BICON may be controlled through the use of firewall 1210, 1215 and RSA Dynamics SecurID with Radius database technology 1200, 1205 which provides physical access token technology. In one embodiment, every individual at each originating member who may use BINET, has an individual logon identity, and must be in possession of his/her unique SecureID token. The use of generic IDs or broker-wide IDs may be contractually prohibited by a broker-to-broker network operator in its agreements with customers.

All data packets sent over BICON may be encrypted using 128 bit SSL technology so as to ensure complete privacy and confidentiality of the information.

BICON is available in several different physical implementations depending on the volume and response characteristics required by the originating member.

In addition if and when the broker-to-broker system's customers use the value-added features of the broker-to-broker system service, e.g. interactive voice messaging, they will need to upgrade the communications bandwidth to minimize adverse impact to the transaction components. A major consideration in the implementation of BICON is to minimize propagation or latency delays.

For occasional or light usage, i.e. a single BIWEB session a 56K V90 modem dial-up session should suffice. Dial-up connections may be used almost always to initially connect the broker-to-broker systems customers into BINET, while higher capacity is being installed. Dial-up connections will also be of use to those originating members who have a field force, mobile workers, and/or very small outlying branch offices. In addition, originating members may most probably use dial-up connections in the less well developed countries where order volume will be small, and the telecommunications infrastructure is not developed and/or slow to respond.

For more demanding light usage, i.e. 2 BIWEB sessions, a 56/64Kbit ISDN dial-up session should suffice. ISDN or high bit rate digital interfaces for dial-up connections may be seen as the preferable alternative to analogue dial-up where the telecommunications infrastructure allows and/or the break-even point between dial-up and fixed line economics has not been reached.

For those originating members who choose to implement computer connections into BINET using

BIAPI, it is expected that different characterizations will apply which will be specific to that originating member's needs and expectations.

SELCON is the physical networking infrastructure that connects the fulfilling members with the broker-to-broker system processing core. In one modification SELCON is a Virtual Private Digital Network, VPDN 1320. SELCON is a physically secure, and information secure, guaranteed quality service, QOS, data communications network.

SELCON is installed between the fulfilling member's location and the nearest the broker-to-broker system POP (point of presence). In another modification SELCON comprises the internet.

Access to SELCON may be controlled through the use of RSA Dynamics SecurID 1300, 1305 which provides physical access token technology. Every individual at each fulfilling member who will use SELNET, must have an individual logon identity, and must be in possession of his/her unique SecureID token. The use of generic IDs or broker-wide IDs may be contractually prohibited by the broker-to-broker network operator in its agreements with the broker-to-broker system's customers.

All data packets sent over SELCON may be encrypted using 128 bit SSL technology so as to ensure complete privacy and confidentiality of the information.

SELCON is available in several different physical implementations depending on the volume and response characteristics required by the fulfilling member. In addition if and when broker-to-broker's customers use the value-added features of the broker-to-broker service, e.g. interactive voice messaging, they will need to upgrade the communications bandwidth to minimize adverse impact to the transaction components. A major consideration in the implementation of SELCON is to minimize propagation or latency delays.

For occasional or light usage, i.e. a single SELWEB session, a 56K V90 modem dial-up session should suffice. Dial-up connections may be used almost always to initially connect the broker-to-broker system 's customers into SELNET, whilst higher capacity is being installed. In addition, fulfilling members will most probably use dial-up connections in the less well developed countries where order volume will be small, and the telecommunications infrastructure is not developed and/or slow to respond.

For more demanding light usage, i.e. 2 SELWEB sessions, a 56/64Kbit ISDN dial-up session should suffice. ISDN or other high bit rate digital dial-up connections will be seen as the preferable alternative to analogue dial-up where the telecommunications infrastructure allows and/or the break-even point between dial-up and fixed line economics has not been reached.

For higher volumes, improved response times, and where on-line times cross the classic break-even point between dial-up and fixed-line, fixed or leased lines will need to be installed. The minimum fixed-line speed available for the broker-to-broker system may be 64Kbits/second. Higher speeds will need to be installed in proportion to the number of concurrent the broker-to-broker system sessions in use at the fulfilling member's location, and the response times which they wish to achieve. As a rule of thumb, 2 SELNET sessions can be concurrently handled per 64Kbit/sec segment when using just SELWEB. For those fulfilling members who choose to implement computer connections into SELNET using SELAPI, it is expected that different characterizations will apply which will be specific to that originating broker's needs and expectations.

SUMMARY

Thus, preferred embodiments provide an order delivery and management facility for use by members to communicate with each other and to their respective settlement agents. The system includes an integrated network of applications through which a member seeking fulfillment of a securities transaction order may enter the order into the system for routing to another designated member who is a broker qualified to fulfill the order or part thereof on the basis of prevailing conditions in the fulfilling member's market, subject to conditions specified by the originating member. Nothing inherent in the system itself brings together orders or trading interests. The parties using the system may be limited to originating members, fulfilling members and settlement agents. Brokers or for example, investment banks may use the system as originating or fulfilling members or both. Investors need not have access to the system other than through members.

Originating members using the system are charged a flat annual fee for up to a specified number of orders that result in the generation by the system of settlement instructions upon execution of a trade, based on an estimate of anticipated usage of the system by each such broker. The annual fee may vary according to a specific member's discount schedule or system-wide transaction volume schedules. A relatively nominal excess usage fee may be charged only if the originating member's use of the system generates a number of settlement instructions in excess of the applicable annual fee limit, in order to compensate the network operator for the additional use of the system's transmission and processing capabilities. Similarly, the network operator may assess excess usage fees to fulfilling members who generate more than a fixed number of settlement instructions through the system on orders sent to them through the system that they have executed. Although excess usage fees are ultimately calculated based on the total number of executed trades processed to the settlement subsystem, as a means of measuring usage of the system, the receipt of such fees are not dependent on the completion of transactions as no rebate are paid if a trade submitted to settlement does not actually close absent a failure of the system. If a transaction is re-run through the settlement subsystem due to a system failure or error, a credit may be accorded to the party charged.

The system does not change the sequencing of orders received from the originating member, nor prioritizes orders from any given originating member, nor among originating members. The system does not change the sequencing of fulfillment advisories transmitted from a fulfilling member back to an originating member.

Agreements between the network operator and members contracting to use the system may require that brokers have and maintain registration or qualification as broker-dealers to the extent required in the respective jurisdictions in which they conduct business, including the registration of foreign brokers whose activities in the United States or with U.S. investors require registration as broker-dealers under the Exchange Act. Brokers using the system may receive no warranty or guarantee from the network operator concerning execution quality in the fulfillment of orders.

In certain modified embodiments it is expected that OMIF 202 may apply a significant amount of sensibility checking to the order, which in the limit may be as rigorous as the checks applied by broker-to-broker processing system 200, but the intent of the OMIF checks are mainly to provide the order originator with a more rapid response time.

It will be apparent that certain alternative embodiments of the broker-to-broker retain many of the advantages of the preferred embodiments. Specifically, a preferred broker-to-broker system enables originating members to send orders for execution to fulfilling members using a secure connection. The resulting transactions can be cleared and settled in the name of the two parties using the broker-to-broker system who are able to manage the process from end terminals.

The broker-to-broker system enables originating members to enter orders for transmission to the selected fulfilling members.

The broker-to-broker system enables fulfilling members to receive order requests from the originating members.

The broker-to-broker system enables fulfilling members to enter fill/execution reports for transmission back to the originating member.

The broker-to-broker system enables the originating member to receive fill/execution advisories/reports.

The broker-to-broker system enables either party to review the status of their orders.

The broker-to-broker system enables either party to request to modify attributes of the order.

The broker-to-broker system enables the originating member and fulfilling member to receive modification requests and to accept or reject the request.

The broker-to-broker system enables either party to request to cancel an order.

The broker-to-broker system enables either party to receive cancellation requests and to accept or reject the request.

The broker-to-broker system enables either party to sign off an order, *i.e.* initiate the settlement process for the fills/executions against his/her order.

The broker-to-broker system enables the OM and FM to receive sign off requests and to accept or reject the request.

The broker-to-broker system provides settlement management capability for those broker-to-broker system customers who wish to contract out that service to the broker-to-broker system. In this instance, broker-to-broker system will generate and control all messaging to and from the customer's elected agent bank and custodian in the name of broker-to-broker system customer.

The broker-to-broker system enables either party to upload material, e.g. notes, pricing models/tools/guidelines, audio clips, video clips to central electronic information library.

The broker-to-broker system enables either party to search the electronic information library and download material.

The broker-to-broker system enables either party to interrogate their billing account with broker-to-broker system to obtain information such as: transaction fees outstanding to broker-to-broker system; customer's membership fees outstanding;

rebates applicable; agreed fee structure; and average TPD for week, month, quarter.

The broker-to-broker system enables either party to send messages directly to other broker-to-broker system customer(s) using the secure system network facilities.

The broker-to-broker system enables either party to obtain information about their use of broker-to-broker's facilities such as: transaction history by: time period, security, market, exchange, sector, geographical region, execution quality statistics, VWAP, security control, authorized names, and passwords.

The broker-to-broker system enables either party to search for security information held within the databases of the broker-to-broker system.

The broker-to-broker system enables either party to search for member information held within the databases of the broker-to-broker system.

Having described in detail embodiments of the present invention, the following rules, legalities and business principles may be applied.

TYPE OF ENTITY

Preferred embodiments of the broker-to-broker system do not function as a broker, bank, clearing house, exchange, crossing network, auction room, aggregator or other place where the change of beneficial ownership of securities takes place. Other embodiments may however include one or more of the above or other functions. The network provides a conduit through which orders can be transmitted for execution; described embodiments of broker-to-broker system do not provide the capability to match buyer and seller within it.

BANK AND SECURITIES ACCOUNTS

Broker-to-broker systems preferably never take ownership of stock, or monies in the transaction. Embodiments of the broker-to-broker system thus need not have securities account(s) and shall not settle securities for its "own account." In order to prevent any confusion or, aspersion of front running or other infringement of securities regulations, people closely associated with the embodiments described herein, e.g. employees, their direct families, consultants, must conduct their investment activities using a recognized and regulated brokerage, and must submit copies of their brokerage transaction records, and must seek prior approval from the broker-to-broker system Compliance officer so that the broker-to-broker system can protect its reputation, assure its customers that confidentiality is being observed, and assure the regulators that the broker-to-broker system has put in place governance to prevent abuse. (i.e. although the broker-to-broker system is not a regulated entity, it conducts and operates its business under the same if not more rigorous set of rules as implemented by leading financial institutions).

In preferred embodiments, the broker-to-broker system does have a bank account(s), which exist purely for the purposes of paying for goods, services, taxes and other expenses, and for receiving payment for services rendered to its members and other income.

ROLE IN TRANSACTIONS

A preferred broker-to-broker system does not take any commissions, or spreads on transactions the broker-to-broker system's only financial involvement in a transaction is the levy of a specific fees see section Transaction Fees below.

TRANSACTION FEES

Preferred broker-to-broker systems charge an annual membership fee, with additional fixed fees for orders routed through the system and surcharges for settlement support. The originating members are responsible for all additional fees and surcharges other than those surcharges for mistakes made by the fulfilling member.

TRANSACTION SEQUENCING

Preferred embodiments of the broker-to-broker system do not change the sequencing of orders received from the originating member, i.e. there is no prioritization of orders from any given originating member, nor is there any prioritization of orders amongst originating members; ALL members and their orders are treated equally.

Broker-to-broker systems preferably do not change the sequencing of execution/fill advisories transmitted from the fulfilling member back to the originating member.

REGULATORY POSITION

Since preferred embodiments of the broker-to-broker system seek always to be regulatory and business neutral, they do not oblige members to carry out any action which may offend or otherwise contravene the rules of their regulatory bodies or in any way influence the investment and execution decisions of its members. In particular, broker-to-broker system will not solicit orders or give financial advice and members of the system have no obligation to deal with any other member(s). To that extent, members of the broker-to-broker system shall have no warranty or guarantee from the broker-to-broker system concerning execution quality or service obtained by members from other members.

BEST EXECUTION

In most jurisdictions the fulfilling broker of a securities exchange is under a regulatory obligation to provide "Best Execution", i.e. deliver the best price possible, to his/her customer taking into account all the factors surrounding the order, and the state of the market; specific factors commonly important to determining how to "best execute" an order are related to the size of the order compared with the normal market trading size so as to minimize market impact, the urgency with which the customer wants the order filled, e.g. the customer may be willing to pay a premium to the prevailing market price if his/her order can be filled in its entirety in one immediate action rather than having to have the order worked over the day where he/she is at risk that the price might rise/fall due to some exogenous event. In some markets transparency of transactions is limited, thus in order to enable the originating members to exercise more control over the quality of their executions, the broker-to-broker system provides at least two mechanisms:

The originating member can designate the execution venue which they require the fulfilling member to use when fulfilling the order, e.g. on exchange, off inventory, cross, best execution. In most jurisdictions the fulfilling member is released from his/her obligation to obtain "best execution" when their customer specifically demands a particular method of execution. The broker-to-broker system order entry front-end allows the originating member to specify and set the default execution venue for each market and member.

Every fill reported back by the fulfilling member is time-stamped and the best official bid and offer as published at that time by that fulfilling member's market authority, (where available), is automatically appended to the fill advisory. This information enables the originating members to compare the execution quality, which they have received from their fulfilling member vs. the pricing in the market. (Ideally the price stamped on the fill should take the best bid and offer available for the number of securities in the fill, e.g. if the official best bid and offer is for 1,000 shares, but the fill is for 5,000 shares then the best bid and offer are priced incorrectly. This capability to more precisely price stamp the fills is available in the more transparent, more developed, and usually electronic markets which to some extent reduces the value of the price stamp to the originating member from a "best execution" perspective, since such markets are typically also more rigorously regulated.)

REGULATORY REPORTING

On request the broker-to-broker system may on behalf of the parties to the transaction generate and transmit reports to the regulators of each party to the transaction. Such reports are designated in the name of the entity on whose behalf the broker-to-broker system has generated and transmitted the message.

SELECTING THE FULFILLING BROKERS

Preferred embodiments of the broker-to-broker system do not prioritize amongst fulfilling members. All orders from the originating member should fully designate the identity of the desired fulfilling member. The broker-to-broker system provides facilities to enable the originating member to select amongst the universe applicable for a given security/market for their desired fulfilling member and then to designate the selected fulfilling member as their default selection for all subsequent orders for that security/market until changed by the originating member. Preferred embodiments thus cannot select the fulfilling member on behalf of the originating member. The broker-to-broker system described herein does not change the identity of the originating member's selected fulfilling member on an order. Where an originating member's selection of fulfilling member is invalid, the originating member is prompted to choose a valid fulfilling member before the order are further processed by the broker-to-broker system.

TRANSACTION STRUCTURE

The two parties to a transaction across the broker-to-broker system network are the initiating member (in this case the member who enters an order into the system, i.e. the originating member), and the fulfilling member (in this case the member who fills the order, i.e. the fulfilling member). The legality/contract of the transaction is as follows:

Let A be the originating member issuing a buy stock

Let B be the fulfilling member fulfilling the buy order

Then the transaction, which takes place, is:

B sells stock to A

and

A buys stock from B

Note At no time does this embodiment of the broker-to-broker system enter into the transaction, thus concepts such as riskless principal, agency, principal are completely excluded from the broker-to-broker system business, legal, and financial processes.

CONFIDENTIALITY

Preferably, elements of the broker-to-broker system do not know the identity of the entity/person on whose behalf the originating member entered the order. Any reference code attached to the order by the originating member are preserved and attached to all messages relating to that order and its subsequent executions, however embodiments of the broker-to-broker system need not translate/map that reference code or in any other way alter such reference code. It is possible that the originating member may need to attach such reference code to enable them within their systems to be able to determine the eventual beneficiary of the proceeds of the transactions. The broker-to-broker system need not store or otherwise record such reference codes and need not use the presence of or omission of such reference codes for any purpose whatsoever.

In addition, preferred embodiments of the broker-to-broker system do not know the identity of the entity from/to which the fulfilling member bought/sold or otherwise obtained the securities in order to fulfil the originating member's order.

PREFERRED SETTLEMENT ARRANGEMENTS

Settlement of the trade between the originating member and their customer is within the purview of the originating member. For example, the price charged to the originating member's customer, fees, quantity of shares received/delivered, date of delivery, where delivered are at the originating member's discretion. The broker-to-broker system need not generate settlement instructions, confirmations or other forms of advisories, messages to the originating member's customer, unless requested to by the originating member.

Settlement of the trade between the fulfilling member and the "street" is within the purview of the fulfilling member. The broker-to-broker system can generate settlement instructions, confirmations or other forms of advisories, and messages to the fulfilling member's "street" counterparties.

The broker-to-broker system may on request and on behalf of either or both parties to the transaction generate and transmit settlement instructions, designated and in their name to their respective agent banks and custodians. In this case, the preferred broker-to-broker system issues the following set of settlement instructions:

Let A be the originating member issuing a buy order for shares of stock, Z, quantity of shares = Y

Let B be the fulfilling member receiving the order from A

Let the price of the execution be X Euros

Let the commission charged by the fulfilling member be C%

Let the local taxes and other regulatory charges for the fulfilling member's jurisdiction be T%

Party B instructions

Party B will deliver Y shares of security Z to Party A

Party B will receive monies from party A calculated in general as follows

(X * Y) + [(X * Y)*C%] + [(X * Y)*T%] = TOTAL CONSIDERATION

In related embodiments, jurisdictional rules and conventions are taken into account in the calculation method shown above to allow for the different ways in which taxes and other fees are levied in the different markets.

Party A Instructions

Party A will receive Y shares of security Z

Party A will deliver monies to party B calculated in general as follows:

(X * Y) + (X * Y)*C% + (X * Y)*T%

In related embodiments, jurisdictional rules are conventions are further catered for in the calculation method shown above to allow for the different ways in which taxes and other fees are levied in the different markets.

Party A will deliver monies to the broker-to-broker system calculated as follows

A fixed fee if the total number of transactions processed by PARTY A exceeds the number of transactor's budget in Party A's annual fee.

The fixed transaction fee is purely exemplary and may be varied according to, for example, per originating member discount schedules, per originating member rate, the broker-to-broker system

system-wide per transaction volume related pricing schedules, the broker-to-broker system system-wide per transaction pricing schedules based on geography/market but in general the fee payable will in no way be calculated off or in any other way be related to the value of the security transaction itself.

ACKNOWLEDGEMENTS AND ADVISORIES

On request, the broker-to-broker system may on behalf of either or both parties to the transaction generate trade confirmations or advisories to either or both parties to the transaction. Thus for example on behalf of the fulfilling member embodiments of the broker-to-broker system generates a telex confirming to the originating member the details of the transaction prior to settlement processing so as to permit the originating member's settlement function to confirm that the transaction is as expected. Such acknowledgements / advisories are designated in the name of the entity on whose behalf the broker-to-broker system has generated and transmitted the message. Such acknowledgements / advisories include the text similar to the following: "The information contained within this acknowledgement and the terms and conditions applicable at the time of execution were agreed solely between you and the fulfilling / originating member named below as a result of one or more interactions across the broker-to-broker network. Any disagreement with the following should be notified immediately to the fulfilling / originating member".

PROPRIETARY TRADING

The broker-to-broker application need never know the "investor" details. Therefore the use of the broker-to-broker network to carry out proprietary/own account trading activities by the originating member for the benefit of the originating member itself have no significance to the way the broker-to-broker system processes or otherwise acts upon the order and any subsequent transaction resulting from that order.

CANCELS/AMENDMENTS

At any time in the transaction process the originating member may alter/cancel any part or all of the order. Once an order has been accepted by the designated fulfilling member, any alterations/cancellations to that order become requests to alter/cancel which must be accepted by the designated fulfilling member. In which case any securities purchased/sold by the fulfilling member at the time the alteration/cancellation request was received must be received/delivered to the originating member, i.e. settlement takes place if any executions have taken place prior to the fulfilling member receiving and accepting the change request. That is, partial fills are still contractually binding.

ORDER PRE-REQUISITES

Before an order is accepted by preferred embodiments of broker-to-broker processing system 200 for onward routing to the fulfilling member, the following minimum set of conditions have to be satisfied, i.e. present on the "order form":

a) The security identifier is valid

b) The number of securities to purchase/sell has been specified

c) The direction of the order has been specified, i.e. buy, and sell

d) The price conditions on the order are valid for that specific market and that specific security, e.g. market, limit

e) The duration and timing of the order are valid for the specific security and market, e.g. good till end of day

f) The trade date for the transaction has been suggested

g) The settlement date for the transaction has been suggested

h) The designated fulfilling broker for the order is valid for the security and market

i) The commission rate to be charged by the FM has been suggested

j) The desired execution venue has been selected by the originating member

k) The originating member may select "best execution" as the venue, in which case the fulfilling member may use their discretion as to how to best fulfil the order.

1) The identity of the issuing person at the originating member has been positively authenticated and is valid.

APPENDIX

The Appendix recites Java code used to implement the preferred embodiment described herein with reference to Figures 1 to 9.

APPENDIX

OrderManagerBean.java

```
//
// Description:  This class manages the interface between both online users or
// electronic interfaces to the order management system, with regard to orders.
// The interface includes all actions that can be associated
// with an order.
//
//Package
package com.sapient.ordermanager;
//EJB session imports
import javax.ejb.SessionContext;
import javax.ejb.SessionBean;
// EJB naming and lookup imports
import javax.naming.InitialContext;
import javax.naming.Context;
import java.util.Hashtable;

//Exception Classes
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import javax.naming.NamingException;
import com.sapient.exception.B2BException;
import com.sapient.exception.LoggedException;

//Property Files
//import java.util.Properties;

// Helper classes
import com.sapient.helper.B2BConstants;
import com.sapient.helper.Self;
import com.sapient.helper.StatusReturn;

// Framework class
import com.sapient.framework.ejb.SessionBeanAdapter;
import com.sapient.framework.ejb.common.HomeFactory;
import com.sapient.framework.ejb.common.HomeFactoryException;
import com.sapient.framework.logging.Logger;
import com.sapient.framework.config.ConfiguratorFinder;
import com.sapient.framework.config.PropertyNotFoundException;
```

```
//Order bean
import com.sapient.order.Order;
import com.sapient.order.OrderHome;
import com.sapient.order.OrderPK;
import com.sapient.order.OrderRequestElements;
import com.sapient.order.OrderDetailElements;

import com.sapient.audit.AuditLogProxy;
import com.sapient.audit.Action;

//Order Interface
import com.sapient.interfaces.OrderGTEIF;

import java.util.Date; //Not needed... just used for testing!



/**
 * @author Darach Ó Braonáin
 * @author Sapient Ltd
 * @version 1.0
 * @stereotype SessionBean
 * @remoteInterface com.sapient.ordermanager.OrderManager
 * @homeInterface com.sapient.ordermanager.OrderManagerHome
 */
public class OrderManagerBean extends SessionBeanAdapter
{
    private transient SessionContext context;
    private Order remoteOrderBean = null;

    private static final String thisClass = "OrderManagerBean.";
      //private static final String aTrueInt = "0";
    /**
      * Sets the context of the bean
      * @param context The Bean's Context
      */
    public void setSessionContext(SessionContext context)
    {
      this.context = context;
    }

    /**
```

```
  * This method is called when the container picks this session object
  * and assigns it to a specific session object. Insert code here to
  * acquire any additional resources that it needs when it is in the
  * ready state.
  */
public void ejbActivate()
{
}

/**
  * This method is called when the container diassociates the bean
  * from the session object identity and puts the instance back into
  * the pool of available instances. Insert code to release any
  * resources that should not be held while the instance is in the
  * pool.
  */
  public void ejbPassivate()
  {
  }

  /**
    * This method is invoked when a client invokes the matching create()
    * on the home interface.
    */
  public void ejbCreate()
  {
  }

  /**
    * The container invokes this method in response to a client-invoked
    * remove request. The bean's representation is removed from the
    * container.
    */
  public void ejbRemove()
  {
  }


 /**
  * This method is used to enter a new order. Then it is called, it must be
 * passed a StatusObject, a Self object and an OrderDetailElements object. When
 * it is called, it will call the EnterOrder method on the OrderInterface. This
 * will enter the order in the GTE and then return an order number. It will
 * then call the SubmitOrder method on the orderInterface which will then call
 * the same method on the GTE. If this is successful, this method will then
 * create a new order on the OrderBean using the orderReference that has been
 * returned.
 *
 * @param   StatusReturn                     This object is used to pass
 *                                           status data from method to
 *                                           method to indicate success or
 *                                           failure of certain processes.
 * @param Self                               This object contains the
 *                                           UserId and BrokerId of the
 *                                           person who has logged into the
 *                                           system.
 * @param OrderDetailElements                The OrderDetailElements will
 *                                           contain all the order data
 *                                           that has been entered by the
 *                                           user as well as enriched
 *                                           information from the GTE such
 *                                           as the order number.
 *
 * @exception  none
 * @return     void
 */
 public void createNewOrder(StatusReturn theStatusReturn, Self theSelf, OrderDetailElements
theOrderDetailElements) throws B2BException, RemoteException {
        final String thisMethod = thisClass + "createNewOrder: ";

        // Debug logging to ensure that methods are being called correctly
        Logger.log(Logger.ENTRY, thisMethod + "Entered by: " + theSelf);

        // Ensure the LastActionBrokerId is set
 theOrderDetailElements.setLastActionBrokerId(theSelf.getBrokerId());

        // enterOrder() returns the new order reference.
        try {
            (new AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.CREATE_NEW_ORDER,null,(new
OrderGTEIF()).enterOrder(theOrderDetailElements));
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
```

```
}

/**
 * This method is used to withdraw a pending request. It firstly calls the
 * WithdrawRequest on the OrderInterface. If this call is successful, this
 * method will then access the the OrderBean and call the WithdrawRequest
 * method which will clear the request fields and recalculate the order status.
 *
 * @param StatusReturn                        This object is used to pass
 *                                            status data from method to
 *                                            method to indicate success or
 *                                            failure of certain processes.
 * @param Self                                This object contains the
 *                                            UserId and BrokerId of the
 *                                            person who has logged into the
 *                                            system.
 * @param OrderRequestElements                The OrderRequestElements will
 *                                            contain all the order data
 *                                            that has been entered by the
 *                                            user.
 * @exception  none
 * @return     void
 */
 public void withdrawPendingRequest(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
        final String thisMethod = thisClass + "withdrawPendingRequest: ";

        // Debug logging to ensure that methods are being entered correctly
        Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf);

        String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
        OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

        // Ensure the LastActionBrokerId is set
 theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

        (new OrderGTEIF()).withdrawRequest(theOrderRequestElements);

        try {
```

```
                    (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.WITHDRAW_PENDING_REQUEST,null,theOrderRequestElements.getOrd
erId());
            } catch (CreateException ex) {
                throw new LoggedException("CreateException caught: "+ex);
            }
    }


    /**
     * This method is used to accept an order. It calls the method AcceptOrder on
     * the orderInterface. If this is successful, it calls the acceptOrder method
     * on the OrderBean.
     *
     * @param StatusReturn                        This object is used to pass
     *                                            status data from method to
     *                                            method to indicate success or
     *                                            failure of certain processes.
     * @param Self                                This object contains the
     *                                            UserId and BrokerId of the
     *                                            person who has logged into the
     *                                            system.
     * @param orderDetailElements                 The OrderDetailElements will
     *                                            contain all the order data
     *                                            that has been entered by the
     *                                            user as well as enriched
     *                                            information from the GTE such
     *                                            as the order number.
     * @exception  RemoteException
     * @return     void
     */
     public void acceptOrder(StatusReturn theStatusReturn, Self theSelf, OrderDetailElements
theOrderDetailElements) throws B2BException, RemoteException {
           final String thisMethod = thisClass + "acceptOrder:   ";

           Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf);

           String myOrderReferenceNumber = theOrderDetailElements.getOrderId();
           OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

     // Ensure the LastActionBrokerId is set
     theOrderDetailElements.setLastActionBrokerId(theSelf.getBrokerId());
```

```
        (new OrderGTEIF()).acceptOrder(theOrderDetailElements);

        try {
            (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.ACCEPT_ORDER,null,theOrderDetailElements.getOrderId());
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
    }


    /**
     * This method is used to reject an unaccepted order. Firstly, the method
     * RejectOrder is called on the orderInterface. If this call is successful, the
     * method RejectOrder is called on the OrderBean.
     *
     * @param myStatusReturn                         This object is used to pass
     *                                               status data from method to
     *                                               method to indicate success or
     *                                               failure of certain processes.
     * @param mySelf                                 This object contains the
     *                                               UserId and BrokerId of the
     *                                               person who has logged into the
     *                                               system.
     * @param OrderDetailElement                     The OrderDetailElements will
     *                                               contain all the order data
     *                                               that has been entered by the
     *                                               user as well as enriched
     *                                               information from the GTE such
     *                                               as the order number.
     * @exception  none
     * @return     void
     */
    public void rejectOrder(StatusReturn theStatusReturn, Self theSelf, OrderDetailElements
theOrderDetailElements) throws B2BException, RemoteException {
        final String thisMethod = thisClass + "rejectOrder: ";

        Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

        String myOrderReferenceNumber = theOrderDetailElements.getOrderId();
        OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);
```

```
          // Ensure the LastActionBrokerId is set
     theOrderDetailElements.setLastActionBrokerId(theSelf.getBrokerId());

          (new OrderGTEIF()).rejectOrder(theOrderDetailElements);

          try {
               (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REJECT_ORDER,null,theOrderDetailElements.getOrderId());
          } catch (CreateException ex) {
               throw new LoggedException("CreateException caught: "+ex);
          }
     }

    /**
     * This method is used to cancel an unaccepted order. The cancellation is sent
     * to the orderInterface. This method in turn passes this cancellation on to
     * the GTE. If this is successful, the method CancleOrder will be called on the
     * OrderBean.
     *
     * @param StatusReturn                           This object is used to pass
     *                                               status data from method to
     *                                               method to indicate success or
     *                                               failure of certain processes.
     * @param Self                                   This object contains the
     *                                               UserId and BrokerId of the
     *                                               person who has logged into the
     *                                               system.
     * @param OrderDetailElements                    The OrderDetailElements will
     *                                               contain all the order data
     *                                               that has been entered by the
     *                                               user as well as enriched
     *                                               information from the GTE such
     *                                               as the order number.
     * @exception  none
     * @return     void
     */
     public void cancelOrder(StatusReturn theStatusReturn, Self theSelf, OrderDetailElements
theOrderDetailElements) throws B2BException, RemoteException {
          final String thisMethod = thisClass + "cancelOrder: ";

          Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf);
```

```
        String myOrderReferenceNumber = theOrderDetailElements.getOrderId();
        OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

        // Ensure the LastActionBrokerId is set
    theOrderDetailElements.setLastActionBrokerId(theSelf.getBrokerId());

        // Ignoring returned request reference
        (new OrderGTEIF()).requestCancelUnacceptedOrder(theOrderDetailElements);

        try {
            (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.CANCEL_ORDER,null,theOrderDetailElements.getOrderId());
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
    }


    /**
     * This method is used to request the cancellation of an accepted order. The
     * request is sent to the orderInterface. This method in turn passes this
     * cancellation request on to the GTE. It does this by calling
     * RequestCancelOrder(...) method. If this is successful, the method
     * requestCancel will be called on the OrderBean.
     *
     * @param StatusReturn                         This object is used to pass
     *                                             status data from method to
     *                                             method to indicate success or
     *                                             failure of certain processes.
     * @param Self                               This object contains the
     *                                             UserId and BrokerId of the
     *                                             person who has logged into the
     *                                             system.
     * @param OrderRequestElements                 The OrderRequestElements will
     *                                             contain all the order data
     *                                             that has been entered by the
     *                                             user.
     * @exception  none
     * @return     void
     */
```

```
      public void requestCancelOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
            final String thisMethod = thisClass + "requestCancelOrder: ";

            Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

            String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
            OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

            // Ensure the LastActionBrokerId is set
      theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

            // Ignoring returned request reference
            (new OrderGTEIF()).requestCancelOrder(theOrderRequestElements);

            try {
                  (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REQUEST_CANCEL_ORDER,null,theOrderRequestElements.getOrderId
());
            } catch (CreateException ex) {
                  throw new LoggedException("CreateException caught: "+ex);
            }
      }


      /**
        * This method is used to accept the cancellation of an accepted order. The
        * acceptance is sent to the orderInterface. This method in turn passes this
        * cancellation acceptance on to the GTE. It does this by calling
        * AcceptCancelOrder(...) method. If this is successful, the method
        * acceptCancel will be called on the OrderBean.
        *
        * @param StatusReturn                              This object is used to pass
        *                                                  status data from method to
        *                                                  method to indicate success or
        *                                                  failure of certain processes.
        * @param Self                                      This object contains the
        *                                                  UserId and BrokerId of the
        *                                                  person who has logged into the
        *                                                  system.
        * @param OrderRequestElements                       The OrderRequestElements will
```

```
         *                                               contain all the order data
         *                                               that has been entered by the
         *                                               user.
         * @exception  none
         * @return     void
         */
      public void acceptRequestCancelOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
            final String thisMethod = thisClass + "acceptRequestCancelOrder: ";

            Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

            String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
            OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);
            (new OrderGTEIF()).acceptCancelOrder(theOrderRequestElements);

            // Ensure the LastActionBrokerId is set
      theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

            try {
                  (new AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.ACCEPT_REQUEST_CANCEL_ORDER,
null,theOrderRequestElements.getOrderId());
            } catch (CreateException ex) {
                  throw new LoggedException("CreateException caught: "+ex);
            }
      }

      /**
         * This method is used to reject the cancellation of an accepted order. The
         * rejection is sent to the orderInterface. This method in turn passes this
         * cancellation rejection on to the GTE. It does this by calling
         * RejectModifyCancelOrder(...) method. If this is successful, the method
         * rejectCancel will be called on the OrderBean.
         *
         * @param StatusReturn                            This object is used to pass
         *                                               status data from method to
         *                                               method to indicate success or
         *                                               failure of certain processes.
         * @param Self                                    This object contains the
         *                                               UserId and BrokerId of the
         *                                               person who has logged into the
         *                                               system.
```

```
     * @param OrderRequestElements                 The OrderRequestElements will
     *                                             contain all the order data
     *                                             that has been entered by the
     *                                             user.
     * @exception  none
     * @return     void
     */
    public void rejectRequestCancelOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
        final String thisMethod = thisClass + "rejectRequestCancelOrder: ";

        Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

        String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
        OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

        // Ensure the LastActionBrokerId is set
    theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

        (new OrderGTEIF()).rejectCancelOrder(theOrderRequestElements);

        try {
            (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REJECT_REQUEST_CANCEL_ORDER,null,theOrderRequestElements.get
OrderId());
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
    }


    /**
     * This method is used to request a modification of an accepted order. The
     * request is sent to the orderInterface. This method in turn passes this
     * modification request on to the GTE. It does this by calling
     * RequestModifyOrder(...) method. If this is successful, the method
     * requestModify will be called on the OrderBean.
     *
     * @param StatusReturn                         This object is used to pass
     *                                             status data from method to
     *                                             method to indicate success or
     *                                              failure of certain processes.
     * @param Self                                  This object contains the
     *                                              UserId and BrokerId of the
     *                                              person who has logged into the
     *                                              system.
     * @param OrderRequestElements                  The OrderRequestElements will
     *                                              contain all the order data
     *                                              that has been entered by the
     *                                              user.
     * @exception  none
     * @return     void
     */
    public void requestModifyOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
        final String thisMethod = thisClass + "requestModifyOrder: ";

        Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

        String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
        OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

        // Ensure the LastActionBrokerId is set
    theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

        // Ignoring returned request reference
        (new OrderGTEIF()).requestModifyOrder(theOrderRequestElements);

        try {
            (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REQUEST_MODIFY_ORDER,null,theOrderRequestElements.getOrderId
());
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
    }


   /**
    * This method is used to accept the modification of an accepted order. The
    * acceptance is sent to the orderInterface. This method in turn passes this
    * cancellation acceptance on to the GTE. It does this by calling
    * AcceptModifyOrder(...) method. If this is successful, the method
    * acceptModify will be called on the OrderBean.
    *
    * @param StatusReturn                        This object is used to pass
    *                                            status data from method to
    *                                            method to indicate success or
    *                                            failure of certain processes.
    * @param Self                                This object contains the
    *                                            UserId and BrokerId of the
    *                                            person who has logged into the
    *                                            system.
    * @param OrderRequestElements                The OrderRequestElements will
    *                                            contain all the order data
    *                                            that has been entered by the
    *                                            user.
    * @exception  none
    * @return     void
    */
    public void acceptRequestModifyOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
        final String thisMethod = thisClass + "acceptRequestModifyOrder: ";

        Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

        String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
        OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);
        (new OrderGTEIF()).acceptModifyOrder(theOrderRequestElements);

        // Ensure the LastActionBrokerId is set
    theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

        try {
            (new AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.ACCEPT_REQUEST_MODIFY_ORDER,
null,theOrderRequestElements.getOrderId());
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
    }


   /**
    * This method is used to reject the modification of an accepted order. The
```

```
      * rejection is sent to the orderInterface. This method in turn passes this
      * cancellation rejection on to the GTE. It does this by calling
      * RejectModifyCancelOrder(...) method. If this is successful, the method
      * rejectModify will be called on the OrderBean.
      *
      * @param StatusReturn                               This object is used to pass
      *                                                  status data from method to
      *                                                  method to indicate success or
      *                                                  failure of certain processes.
      * @param Self                                      This object contains the
      *                                                  UserId and BrokerId of the
      *                                                  person who has logged into the
      *                                                  system.
      * @param OrderRequestElements                      The OrderRequestElements will
      *                                                  contain all the order data
      *                                                  that has been entered by the
      *                                                  user:
      * @exception  none
      * @return     void
      */
      public void rejectRequestModifyOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
            final String thisMethod = thisClass + "rejectRequestModifyOrder: ";

            Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

            String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
            OrderPK myorderReferencePK = new OrderPK(myOrderReferenceNumber);

            // Ensure the LastActionBrokerId is set
      theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

            (new OrderGTEIF()).rejectModifyOrder(theOrderRequestElements);

            try {
                  (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REJECT_REQUEST_MODIFY_ORDER,null,theOrderRequestElements.get
OrderId());
            } catch (CreateException ex) {
                  throw new LoggedException("CreateException caught: "+ex);
            }
      }
```

```
    /**
      * This method is used to request sign off of an order. The request for signoff
      * is sent to the orderInterface. This method in turn passes this signoff
      * request on to the GTE. It does this by calling the RequestSignoff(...)
      * method. If this is successful, the method requestSignoffOrder will be called
      * on the OrderBean.
      *
      * @param StatusReturn                              This object is used to pass
      *                                                  status data from method to
      *                                                  method to indicate success or
      *                                                  failure of certain processes.
      * @param Self                                  This object contains the
      *                                                  UserId and BrokerId of the
      *                                                  person who has logged into the
      *                                                  system.
      * @param OrderDetailElements                       The OrderDetailElements will
      *                                                  contain all the order data
      *                                                  that has been entered by the
      *                                                  user as well as enriched
     .*                                                  information from the GTE such
      *                                                  as the order number.
      * @exception  none
      * @return     void
     */
      public void requestSignoffOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
            final String thisMethod = thisClass + "requestSignoffOrder: ";

            Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

            String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
            OrderPK myOrderReferencePK = new OrderPK(myOrderReferenceNumber);

            // Ensure the LastActionBrokerId is set
      theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

            // Ignoring returned request reference
            (new OrderGTEIF()).requestSignOff(theOrderRequestElements);

            try {
                (new AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REQUEST_SIGNOFF_ORDER,
null,theOrderRequestElements.getOrderId());
            } catch (CreateException ex) {
                throw new LoggedException("CreateException caught: "+ex);
            }
        }


    /**
     * This method is used to accept sign off of an order. The acceptance of
     * signoff is sent to the orderInterface. This method in turn passes this
     * signoff acceptance on to the GTE. It does this by calling the
     * AcceptSignoff(...) method. If this is successful, the method
     * AcceptSignoffOrder will be called on the OrderBean.
     *
     * @param StatusReturn                      This object is used to pass
     *                                          status data from method to
     *                                          method to indicate success or
     *                                          failure of certain processes.
     * @param Self                              This object contains the
     *                                          UserId and BrokerId of the
     *                                          person who has logged into the
     *                                          system.
     * @param OrderDetailElements               The OrderDetailElements will
     *                                          contain all the order data
     *                                          that has been entered by the
     *                                          user as well as enriched
     *                                          information from the GTE such
     *                                          as the order number.
     * @exception  none
     * @return     void
     */
     public void acceptRequestSignoffOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
theOrderRequestElements) throws B2BException, RemoteException {
            final String thisMethod = thisClass + "acceptSignoffOrder: ";

            Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

            String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
            OrderPK myOrderReferencePK = new OrderPK(myOrderReferenceNumber);
            // Ensure the LastActionBrokerId is set
        theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

            (new OrderGTEIF()).acceptSignOff(theOrderRequestElements);

            try {
                  (new AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.ACCEPT_SIGNOFF_ORDER,
 null,theOrderRequestElements.getOrderId());
            } catch (CreateException ex) {
                  throw new LoggedException("CreateException caught: "+ex);
            }
        }


      /**
       * This method is used to reject sign off of an order. The rejection of
       * signoff is sent to the orderInterface. This method in turn passes this
       * signoff rejection on to the GTE. It does this by calling the
       * RejectSignoff(...) method. If this is successful, the method
       * RejectSignoffOrder will be called on the OrderBean.
       *
       * @param StatusReturn                           This object is used to pass
       *                                               status data from method to
       *                                               method to indicate success or
       *                                               failure of certain processes.
       * @param Self                                   This object contains the
       *                                               UserId and BrokerId of the
       *                                               person who has logged into the
       *                                               system.
       * @param OrderDetailElements                    The OrderDetailElements will
       *                                               contain all the order data
       *                                               that has been entered by the
       *                                               user as well as enriched
       *                                               information from the GTE such
       *                                               as the order number.
       * @exception  none
       * @return     void
       */
       public void rejectRequestSignoffOrder(StatusReturn theStatusReturn, Self theSelf, OrderRequestElements
 theOrderRequestElements) throws B2BException, RemoteException {
             final String thisMethod = thisClass + "rejectSignoffOrder: ";
        Logger.log(Logger.ENTRY,thisMethod +"Entered by: "+theSelf.toString());

        String myOrderReferenceNumber = theOrderRequestElements.getOrderId();
        OrderPK myOrderReferencePK = new OrderPK(myOrderReferenceNumber);

        // Ensure the LastActionBrokerId is set
    theOrderRequestElements.setLastActionBrokerId(theSelf.getBrokerId());

        (new OrderGTEIF()).rejectSignOff(theOrderRequestElements);

        try {
            (new
AuditLogProxy()).logAction(theSelf.getBrokerId(),Action.REJECT_SIGNOFF_ORDER,null,theOrderRequestElements.getOrderId
());
        } catch (CreateException ex) {
            throw new LoggedException("CreateException caught: "+ex);
        }
    }
    /**
     * This method is used to get all order details. It does this by calling the
     * getOrderDetails method on the OrderBean. This method returns an
     * OrderDetailElements object that contains all the order details. The details
     * that are required can then be extracted from this.
     *
     * @param myStatusReturn                          This object is used to pass
     *                                                status data from method to
     *                                                method to indicate success or
     *                                                failure of certain processes.
     * @param mySelf                                  This object contains the
     *                                                UserId and BrokerId of the
     *                                                person who has logged into the
     *                                                system.
     * @param OrderDetailElements                     The OrderDetailElements will
     *                                                contain all the order data
     *                                                that has been entered by the
     *                                                user as well as enriched
     *                                                information from the GTE such
     *                                                as the order number.
     * @exception  none
     * @return     OrderDetailElements                This variable will contain all
     *                                                the order details. The ones
     *                                                that are required can then be
     *                                  extracted from the object.
     */
    public OrderDetailElements getOrderDetails(Self mySelf, OrderDetailElements theOrderDetailElements) throws
B2BException, RemoteException {
            final String thisMethod = thisClass + "getOrderDetails: ";

            String OrderReferenceNumber = theOrderDetailElements.getOrderId();//this line will be replaced by the
above code
            OrderPK myorderReferencePK = new OrderPK(OrderReferenceNumber);
            OrderDetailElements myOrderDetailElements = new OrderDetailElements();

            try{
                OrderHome orderHome = (OrderHome) HomeFactory.findHome("com.sapient.order.OrderHome");
                remoteOrderBean = orderHome.findByPrimaryKey(myorderReferencePK);
                myOrderDetailElements = remoteOrderBean.getOrderDetails();
            } catch (FinderException ex) {
                throw new LoggedException(thisMethod + "orderId : " +theOrderDetailElements.getOrderId()+"
brokerId : " +mySelf.getBrokerId() + " shadowedBy : "+mySelf.getShadowedBy()+" Could not find order with PK:
["+myorderReferencePK+"]",ex);
            } catch (HomeFactoryException ex) {
                throw new LoggedException(thisMethod + "orderId : " +theOrderDetailElements.getOrderId()+"
brokerId : " +mySelf.getBrokerId() + " shadowedBy : "+mySelf.getShadowedBy()+" Unable to get home factory for
OrderHome",ex);
            }

            Logger.log(Logger.DEBUG,thisMethod + "orderId : " +theOrderDetailElements.getOrderId()+" brokerId : "
+mySelf.getBrokerId() + " shadowedBy : "+mySelf.getShadowedBy()+" "+thisMethod +": has returned the fullowing data:
"+myOrderDetailElements.toString());

            String userBrokerage = mySelf.getBrokerageId();

            Logger.log(Logger.DEBUG, thisMethod + "orderId : " +theOrderDetailElements.getOrderId()+" brokerId : "
+mySelf.getBrokerId() + " shadowedBy : "+mySelf.getShadowedBy()+" "+thisMethod +": User from Brokerage:" +
userBrokerage + " attempting to retrieve order with OB:" + myOrderDetailElements.getOriginatingBrokerageId() + " and
with FB:" + myOrderDetailElements.getFulfillingBrokerageId());

            if ( !((userBrokerage.equals(myOrderDetailElements.getOriginatingBrokerageId())) ||
(userBrokerage.equals(myOrderDetailElements.getFulfillingBrokerageId()))) )
                throw new LoggedException(thisMethod + "orderId : " +theOrderDetailElements.getOrderId()+"
brokerId : " +mySelf.getBrokerId() + " shadowedBy : "+mySelf.getShadowedBy()+" "+thisMethod+"Authentication
exception: The Order: "+myOrderDetailElements.getOrderId()+" could not be displayed!");
```

```
                return myOrderDetailElements;
            }

        }
```

ExecutionManagerBean.java

```
/**
 * Title:        Execution Manager Bean <p>
 * Copyright:    Copyright © 2000 Broker To Broker Networks. All rights reserved. <p>
 * Company:      Sapient <p>
 * @author  Alan Cummins
 * @version 1.0
 *
 * @description  This class manages the interface between both online users or
 * electronic interfaces to the order management system, with regard to order
 * executions (fills). The interface includes all actions that can be associated
 * with an execution (fill).
 */
package com.sapient.executionmanager;

import java.sql.Timestamp;

import javax.ejb.SessionContext;
import javax.ejb.SessionBean;
import javax.ejb.FinderException;
import javax.ejb.CreateException;
import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.rmi.RemoteException;
import java.util.Enumeration;
import java.util.ArrayList;
import com.sapient.framework.sql.GeneralSqlException;
import com.sapient.framework.ejb.common.ConnectionManager;
import com.sapient.framework.ejb.common.ConnectionManagerException;
import com.sapient.framework.config.ConfiguratorFinder;
import com.sapient.framework.config.PropertyNotFoundException;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import com.sapient.framework.util.Day;

import com.sapient.order.Order;
import com.sapient.order.OrderHome;
import com.sapient.order.OrderPK;
import com.sapient.order.OrderDetailElements;

import com.sapient.interfaces.OrderGTEIF;

import com.sapient.order.OrderExecutionElements;
import com.sapient.execution.ExecutionHome;
import com.sapient.execution.ExecutionPK;
import com.sapient.execution.Execution;

import com.sapient.framework.ejb.common.HomeFactory;
import com.sapient.framework.ejb.common.HomeFactoryException;
import com.sapient.framework.logging.Logger;
import com.sapient.helper.B2BConstants;
import com.sapient.helper.Self;
import com.sapient.exception.B2BException;
import com.sapient.exception.LoggedException;
import com.sapient.audit.AuditLogProxy;
import com.sapient.audit.Action;

import com.sapient.framework.ejb.SessionBeanAdapter;

// EJB naming and lookup imports
import java.util.Hashtable;
import java.util.Properties;
import java.util.Calendar;


/**
 * Contains all methods used for execution management of an order.
 * Specifically management of fill actions and display of execution information
 * The associated execution detail elements are contained within a data access object
 * and are associated in a list of executions against an order
 *
 * @stereotype SessionBean
 * @remoteInterface com.sapient.ordermanagement.ExecutionManager
 * @homeInterface com.sapient.ordermanagement.ExecutionManager
 */
public class ExecutionManagerBean extends SessionBeanAdapter
{

    /**
```

```
     * Used for logging purposes
     */
    private static String thisClass = "ExecutionManagerBean.";

      /**
       * This method is invoked when a client invokes the matching create()
       * on the home interface.
       */
      public void ejbCreate()
      {
      }

      /**
       * The container invokes this method in response to a client-invoked
       * remove request. The bean's representation is removed from the
       * container.
       */
      public void ejbRemove()
      {
      }

      /**
     * Enters a fill against the associated order.
     * The order interface is called with all the required information for an execution.
     * The associated order has an execution element set with all the user-entered details.
     * The callback interface will update this order execution with any enriched data at a later stage.
       * @param orderExecutionElement All details relevant for entering a fill
       * @exception     B2BException
       */
      public void enterFill(Self self, OrderExecutionElements anOrderExecutionElement) throws B2BException,
RemoteException {
            // Call the interface and retrieve the execution reference
            Logger.log(Logger.DEBUG,"OrderId : " + anOrderExecutionElement.getOrderId()+"
ExecutionManagerBean.enterFill() called with OrderExecutionsElements: ["+anOrderExecutionElement+"]");
            (new OrderGTEIF()).executionEntry(anOrderExecutionElement);

            try {
                  (new
AuditLogProxy()).logAction(self.getBrokerId(),Action.ENTER_EXECUTION,null,anOrderExecutionElement.getOrderId());
            } catch (CreateException ex) {
                  throw new LoggedException("CreateException caught: "+ex);
            }
    }

    /**
    * Requests the cancellation of an execution associated with an order.
    * The order interface is called to request the cancellation.
    * The order and the associated execution element are also set with the requested
    * cancellation information.
    * @param          anOrderId
    * @param          anExecutionReference
    * @param          aCancelReason
      * @exception    B2BException
      */
    public void requestCancelExecution(Self self, String anOrderId, String anExecutionReference, String
aCancelReason) throws B2BException, RemoteException {
            // Get the details of the relevant execution
            OrderExecutionElements myOrderExecutionElement = getExecution(anExecutionReference);
            myOrderExecutionElement.setExecutionCancelReason(aCancelReason);

            Logger.log(Logger.DEBUG,"OrderId : " + anOrderId+" ExecutionManagerBean.requestCancelExecution() called
with OrderExecutionsElements: ["+myOrderExecutionElement+"], CancelReason: ["+aCancelReason+"]");
            (new OrderGTEIF()).requestCancelExecution(myOrderExecutionElement);

            try {
                  (new
AuditLogProxy()).logAction(self.getBrokerId(),Action.REQUEST_CANCEL_EXECUTION,null,anOrderId);
            } catch (CreateException ex) {
                  throw new LoggedException("CreateException caught: "+ex);
            }
    }

    /**
    * Gets the list of executions against an order.
    * This function is used to return an ArrayList of OrderExecutionElements
    * associated with an Order. Does not return Cancelled executions.
    *
    * @returns   ArrayList                An ArrayList of OrderExecutionElements.
    * @param    orderId
    * @return  ArrayList   A list of all executions associated with an order
    * @exception      B2BException
    */
    public ArrayList displayExecutions(String orderId) throws B2BException, RemoteException {
          try {
                ArrayList array = new ArrayList();
                ExecutionHome executionHome =
(ExecutionHome)HomeFactory.findHome("com.sapient.execution.ExecutionHome");
                Enumeration executions =  executionHome.findByOrderId(orderId);

                while (executions.hasMoreElements()) {
                      Execution execution = (Execution)executions.nextElement();
                      array.add(execution.getOrderExecutionElements());
                }
                return array;
            } catch (HomeFactoryException ex) {
                throw new LoggedException("OrderId : " + orderId+" "+thisClass +
":HomeFactoryException:displayExecutions:" + ex);
            } catch (FinderException ex) {
                throw new LoggedException("OrderId : " + orderId+" "+thisClass +
":FinderException:displayExecutions:" + ex);
            }
      }

    /**
     * Reject the cancellation of a fill.
     * Calls the rejectCancelFill method on the orderInterface.
     * This in turn calls the RejectCancelExecution(...) method on the GTE.
     * If this is successful, the method rejectCancelFill is called on the OrderBean.
     *
     * @param           anOrderId
     * @param           anExecutionReference
     * @param           aRejectionReason
     * @exception B2BException
     * @return      boolean
     */
     public void rejectCancelExecution(Self self, String anOrderId, String anExecutionReference, String
aRejectionReason) throws B2BException, RemoteException {
            // Retrieve all the relevant information for that execution
            OrderExecutionElements myOrderExecutionElement = getExecution(anExecutionReference);
            myOrderExecutionElement.setRejectExecutionCancelReason(aRejectionReason);

            // We need to populate this field, and it doesn't come back in the ExecutionElements info.
            try {
                  OrderHome orderHome = (OrderHome)HomeFactory.findHome("com.sapient.order.OrderHome");
                  Order order = orderHome.findByPrimaryKey(new OrderPK(anOrderId));
                  OrderDetailElements details = order.getOrderDetails();
                myOrderExecutionElement.setRequestReference(details.getRequestReference());
                // XXX: This is a hack! The GTE expects FID_BROKER_ID and FID_USER_ID to be
                // the current actioning user, whereas we use the terms to mean the fulfilling broker.
                // This does work correctly, but the fields should be named more accurately.
                myOrderExecutionElement.setExecutingBrokerageId(details.getOriginatingBrokerageId());
                myOrderExecutionElement.setExecutedBrokerId(details.getOriginatingBrokerId());

            } catch (HomeFactoryException ex) {
                throw new LoggedException("OrderId : " + anOrderId+" "+thisClass +
":HomeFactoryException:rejectCancelExecution:" + ex);
            } catch (FinderException ex) {
                throw new LoggedException("OrderId : " + anOrderId+" "+thisClass +
":FinderException:rejectCancelExecution:" + ex);
            }

            (new OrderGTEIF()).rejectCancelExecution(myOrderExecutionElement);

            try {
                (new AuditLogProxy()).logAction(self.getBrokerId(),Action.REJECT_CANCEL_EXECUTION,null,anOrderId);
            } catch (CreateException ex) {
                throw new LoggedException("CreateException caught: "+ex);
            }
        }

    /**
     * Accept the cancellation of a fill.
     * Calls the acceptCancelFill method on the orderInterface.
     * This in turn calls the AcceptCancelExecution(...) method on the GTE.
     * If this is successful, the method acceptCancelFill is called on the OrderBean.
     *
     * @param           anOrderId
     * @param           anExecutionReference
     * @exception B2BException
     */
      public void acceptCancelExecution(Self self, String anOrderId, String anExecutionReference) throws
B2BException, RemoteException {
            // Find the execution against which we wish to accept the cancellation
            OrderExecutionElements myOrderExecutionElement = getExecution(anExecutionReference);

            // We need to populate this field, and it doesn't come back in the ExecutionElements info.
            try {
                OrderHome orderHome = (OrderHome)HomeFactory.findHome("com.sapient.order.OrderHome");
                Order order = orderHome.findByPrimaryKey(new OrderPK(anOrderId));
                OrderDetailElements details = order.getOrderDetails();

                myOrderExecutionElement.setRequestReference(details.getRequestReference());
                // XXX: This is a hack! The GTE expects FID_BROKER_ID and FID_USER_ID to be
                // the current actioning user, whereas we use the terms to mean the fulfilling broker.
                // This does work correctly, but the fields should be named more accurately.
                myOrderExecutionElement.setExecutingBrokerageId(details.getOriginatingBrokerageId());
                myOrderExecutionElement.setExecutedBrokerId(details.getOriginatingBrokerId());
            } catch (HomeFactoryException ex) {
                throw new LoggedException("OrderId : " + anOrderId+" "+thisClass +
":HomeFactoryException:acceptCancelExecution:" + ex);
            } catch (FinderException ex) {
                throw new LoggedException("OrderId : " + anOrderId+" "+thisClass +
":FinderException:acceptCancelExecution:" + ex);
            }

            (new OrderGTEIF()).acceptCancelExecution(myOrderExecutionElement);

            try {
                (new AuditLogProxy()).logAction(self.getBrokerId(),Action.ACCEPT_CANCEL_EXECUTION,null,anOrderId);
            } catch (CreateException ex) {
                throw new LoggedException("CreateException caught: "+ex);
            }
        }


    /**
     * Status of a request to Cancel a fill
     *
     * <BR><BR><B>Screen Documentation:</B><BR>     FB_STCF_XXX_01, OB_STCF_XXX_01
     *
     * @param       orderID
     * @param           anExecutionReference
     * @exception B2BException
     * @return      orderExecutionElements
     */
    public OrderExecutionElements displayExecution(String anOrderID, String anExecutionReference) throws
B2BException, RemoteException {
            return getExecution(anExecutionReference);
    }
    /**
     * Retreive the execution Element that is pending cancel if any
     * There should always be only fill cancel pending against an order
     * @param     orderId     The order against which there is a pending cancel of fill
     * @return          OrderExecutionElement   The execution against which there is
     *                                                       a cancel fill request
     * @exception B2BException
     * @exception RemoteException
     */
     public OrderExecutionElements retrievePendingCancelFillExecution(String orderId) throws B2BException,
RemoteException {
            try {
                  OrderHome orderHome = (OrderHome)HomeFactory.findHome("com.sapient.order.OrderHome");
                  Order order = orderHome.findByPrimaryKey(new OrderPK(orderId));

                  // This execution reference is the ID of the execution currently being cancelled, if any.
                  String cancelExecutionRef = order.getOrderDetails().getRq_cancelExecutionRef();
                  Logger.log(Logger.DEBUG,"OrderId : " + orderId+" retreivePendingCancelFillExecution() called, it
found: ["+cancelExecutionRef+"]");

                  if (cancelExecutionRef != null) {
                        // Their is a request to cancel an execution outstanding.
                        // Get the execution, and return it.
                        try {
                              ExecutionHome executionHome =
(ExecutionHome)HomeFactory.findHome("com.sapient.execution.ExecutionHome");
                              Execution execution = executionHome.findByPrimaryKey(new
ExecutionPK(cancelExecutionRef));
                              return execution.getOrderExecutionElements();
                        } catch (HomeFactoryException ex) {
                              throw new LoggedException("OrderId : " + orderId+" "+"Couldn't get ExecutionHome:
"+ex);
                        } catch (FinderException ex) {
                              throw new LoggedException("OrderId : " + orderId+" "+"FinderException: Couldn't find
execution reference: ["+cancelExecutionRef+"], exception: "+ex);
                        }
                  } else throw new LoggedException("OrderId : " + orderId+" "+"No execution with request to cancel
fill set against it, for OrderId: ["+orderId+"]");
            } catch (HomeFactoryException ex) {
                  throw new LoggedException("OrderId : " + orderId+" "+"Couldn't get OrderHome: "+ex);
            } catch (FinderException ex) {
                throw new LoggedException("OrderId : " + orderId+" "+"FinderException: Couldn't find order id:
["+orderId+"], exception: "+ex);
            }
        }

        /**
         * Returns an OrderExecutionElements object for the given execution reference.
         * A SQL statement is created and executed and the details of the first record
         * are used to set the attributes of an OrderExecutionElements object which
         * is returned.
         * <p>
         * <b>The function does not currently check for multiple records with the execution
         * reference passed in.</b> - it assumes execution references are unique across all orders.<p>
         *
         * @param   myExecutionReference    The execution reference of the execution.
         *
         * @returns myExecutionElements     OrderExecutionElements containing all the
         *                                              details for that execution.
         */
        public OrderExecutionElements getExecution(String myExecutionReference) throws B2BException {
            try {
                ExecutionHome executionHome =
(ExecutionHome)HomeFactory.findHome("com.sapient.execution.ExecutionHome");
                Execution execution = executionHome.findByPrimaryKey(new ExecutionPK(myExecutionReference));
                return execution.getOrderExecutionElements();
            } catch (HomeFactoryException ex) {
                throw new LoggedException("HomeFactoryException:getExecution:" + ex);
            } catch (FinderException ex) {
                throw new LoggedException("ExecutionManager.getExecution(): No execution with the Execution
Reference [" + myExecutionReference + "] found.");
            } catch (RemoteException ex) {
                throw new LoggedException("RemoteException:getExecution:" + ex);
            }
        }
}
```

CLAIMS

1. A method of operating a computer system to facilitate transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the method comprising:

receiving at a first interface of a processing system at least one information item of an electronic transaction proposal from an originating party;

transmitting at least one information item relating to the electronic transaction proposal from a second interface of the processing system to a fulfilling member;

generating at the processing system settlement criteria to be accepted by the originating party and the fulfilling member; and

receiving from each of the originating party and the fulfilling member an indication of acceptance of the settlement criteria generated by the processing system.

2. A method as in claim 1, wherein the step of receiving the at least one information item of the electronic transaction proposal comprises receiving an information item indicating what is to be transacted.

3. A method as in claim 1 or 2 , wherein the step of receiving the at least one information item of the electronic transaction proposal comprises receiving

an information item identifying a designated fulfilling member to perform the transaction.

4. A method as in any preceding claim, wherein the processing system receives settlement information from said originating party and said fulfilling member.

5. A method as in claim 4, wherein the settlement criteria generated by the processing system are based on settlement information received from the originating party and the fulfilling member.

6. A method as in any preceding claim, wherein the settlement criteria generated by the processing system are based on stored settlement information accessible by the processing system.

7. A method as in any preceding claim, including the step of generating at said processing system settlement instructions on behalf of said originating party and said fulfilling member.

8. A method as in claim 7, wherein settlement instructions are generated responsive to said indications of acceptance of the settlement criteria.

9. A method as in any preceding claim, wherein a fulfilling member requests a modification to at least part of an electronic transaction proposal from an originating party.

10. A method as in any preceding claim, wherein the step of receiving the at least one information item of the electronic transaction proposal comprises receiving information items relating to a proposed transaction selected from one or more of the following:

a transaction type indicator;
a quantity indicator;
a price condition; and
timing information.

11. A method as in claim 10, wherein the timing information indicates a proposed transaction date and/or a proposed settlement date.

12. A method as in any preceding claim, wherein the step of receiving the at least one information item of the electronic transaction proposal comprises receiving information items relating to the proposed transaction from the originating party in two or more sub-steps.

13. A method as in claim 12, wherein an originating party provides a first information item in a first sub-step and the processor system supplies a further relevant information item responsive thereto.

14. A method as in claim 13, wherein the processing system supplies a plurality of further relevant

information items from which the originating party can select.

15. A method as in claim 14, wherein an originating party provides a first information item indicating what is to be transacted in a first sub-step and the processing system prompts the originating party to select from further information items representing exchanges in a further sub-step.

16. A method as in any preceding claim, wherein the processing system converts at least a portion of an electronic transaction proposal from a first format appropriate to the originating party into a second format appropriate to the fulfilling member.

17. A method as in claim 16, wherein the processing system converts a transactional term in a first language appropriate to the originating party into a second language appropriate to the fulfilling member.

18. A method as in claim 16 or 17, wherein the processing system converts a price indication in a first currency appropriate to the originating party into a corresponding price indication in a second currency appropriate to the fulfilling member.

19. A method as in claim 16, 17 or 18, wherein the exchange is a securities exchange and the processing system converts a first security identifier appropriate to the originating party into a second

security identifier appropriate to the fulfilling member.

20. A method as in claim 16, wherein the first format is selected by the originating party.

21. A method as in claim 16, wherein the second format is selected by the fulfilling member.

22. A method as in any preceding claim, wherein the processing system transfers an information item between the first interface and the second interface without altering the format of the information item.

23. A method a in any preceding claim, wherein the processing system generates a predetermined signal to alert a member of a change in status of a proposed transaction.

24. A method according to any preceding claim, wherein a fulfilling member also functions as an originating party by submitting a transaction proposal for execution on a further exchange.

25. A method according to any preceding claim, wherein an originating party also functions as a fulfilling member by performing a transaction on a further exchange.

26. A computer system for facilitating transactions on an exchange, a transaction being performed by a

fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the system comprising:

a first processing system interface adapted to receive one or more information items of an electronic transaction proposal from an originating party;

a second processing system interface adapted to transmit one or more information items relating to the electronic transaction proposal to a fulfilling member;

a processing system for routing communications including information items to and from the first and second interfaces, the processing system being operable to generate settlement criteria to be accepted by the originating party and the fulfilling member, and wherein the processing system is arranged to receive first and second information items indicating acceptance of the settlement criteria by each of the originating party and the fulfilling member.

27. A computer system as in claim 26 wherein the processing system comprises at least one further processing system interface adapted to issue settlement instructions in respect of a transaction performed by the fulfilling member.

28. A computer system according to claim 26 or 27, comprising a data store for holding settlement information.

29. A computer system as in any of claims 26 to 28, wherein the processing system is operable to generate first and second settlement instructions in formats appropriate to settlement agents of the originating party and the fulfilling member.

30. A computer system as in claim 29, comprising a data store for holding said first and second information items indicating acceptance of the settlement criteria by each of the originating party and the fulfilling member.

31. A computer system as in claim 30, wherein the processing system is operable to detect if the originating party and the fulfilling member have accepted the settlement criteria and to generate settlement instructions in response thereto.

32. A computer system according to any of claims 26 to 31, wherein the processing system is operable to generate prompts responsive to receiving an incomplete electronic transaction proposal via said first interface.

33. A computer system according to any of claims 26 to 32, wherein the processing system has access to a data store holding data relevant to transactions and is operable to access the data store in order to provide one or more data items relevant to an electronic transaction proposal.

34. A computer system as in any of claims 26 to 33, comprising a data store for holding conversion information for use in converting at least a portion of an electronic transaction proposal from a first format appropriate to an originating party into a second format appropriate to a fulfilling member.

35. A computer system as in claim 34, wherein the data store holds conversion information selected from one or more of the following:

language conversion information;

currency conversion information; and

conversion information for transaction item identifiers.

36. A computer system as in claim 35, wherein the data store holds conversion information for transaction item identifiers relating to securities.

37. A computer readable medium having stored therein a set of general purpose routines for facilitating transactions on exchanges, the computer readable medium comprising:

a first routine for receiving at least one information item of a transaction proposal from an originating party;

a second routine for transmitting at least one information item of the transaction proposal to a fulfilling member;

a third routine for generating first and second settlement criteria to be accepted by said originating party and said fulfilling member; and

a fourth routine for receiving from each of the originating party and the fulfilling member an indication of acceptance of the settlement criteria.

38. Computer program code for facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to an exchange on behalf of an originating party not having access to the exchange, the program code comprising:

a first set of instructions for receiving at least one information item of a transaction proposal from an originating party;

a second set of instructions for transmitting at least one information item of the transaction proposal to a fulfilling member;

a third set of instructions for generating first and second sets of settlement criteria to be accepted by said originating party and said fulfilling member; and

a fourth set of instructions for receiving from each of said originating party and said fulfilling member an indication of acceptance of the settlement criteria.

39. A method of facilitating transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on

behalf of an originating party not having access to the exchange, the method comprising:

receiving at least one information item of a transaction proposal from an originating party;

transmitting at least one information item of the transaction proposal to a fulfilling member;

generating settlement criteria to be accepted by the originating party and the fulfilling member; and receiving indications of acceptance of the settlement criteria from each of said originating party and said fulfilling member.

40. A method of operating a computer system to facilitate transactions on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the method comprising:

receiving at a first interface of a processing system an information item of an electronic transaction proposal from an originating party, said information item being in a first form appropriate for the originating party;

converting said information item of the electronic transaction proposal from the first form into a second form appropriate to a fulfilling member; and

transmitting the information item in the second form from a second interface of the processing system to a fulfilling member.

41. A computer system for facilitating transactions

on an exchange, a transaction being performed by a fulfilling member having access to the exchange on behalf of an originating party not having access to the exchange, the system comprising:

a first processing system interface adapted to receive an information item of an electronic transaction proposal from an originating party, said information item being in a first form appropriate to the originating party;

a processing system operable to convert the information item of the electronic transaction proposal from the first form into a second form appropriate to a fulfilling member; and

a second processing system interface adapted to transmit the information item in the second form to the fulfilling member.

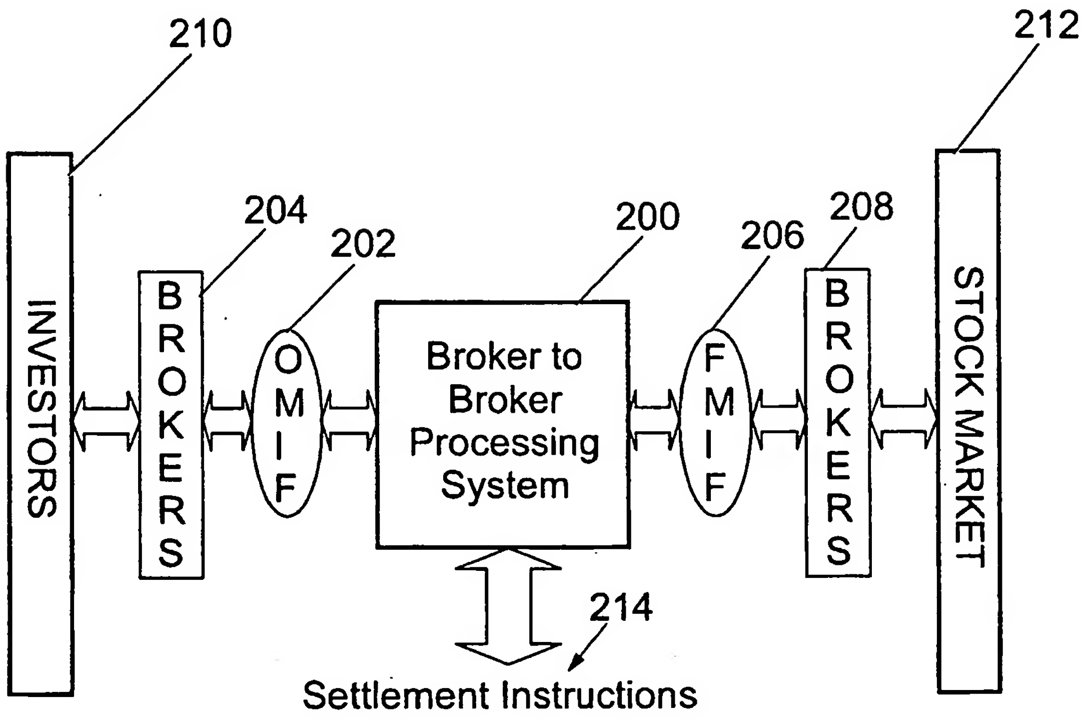
210
212
204
202
200
206
208
INVESTORS
BROKERS
OMIF
Broker to Broker Processing System
FMIF
BROKERS
STOCK MARKET
214
Settlement Instructions

Fig. 1

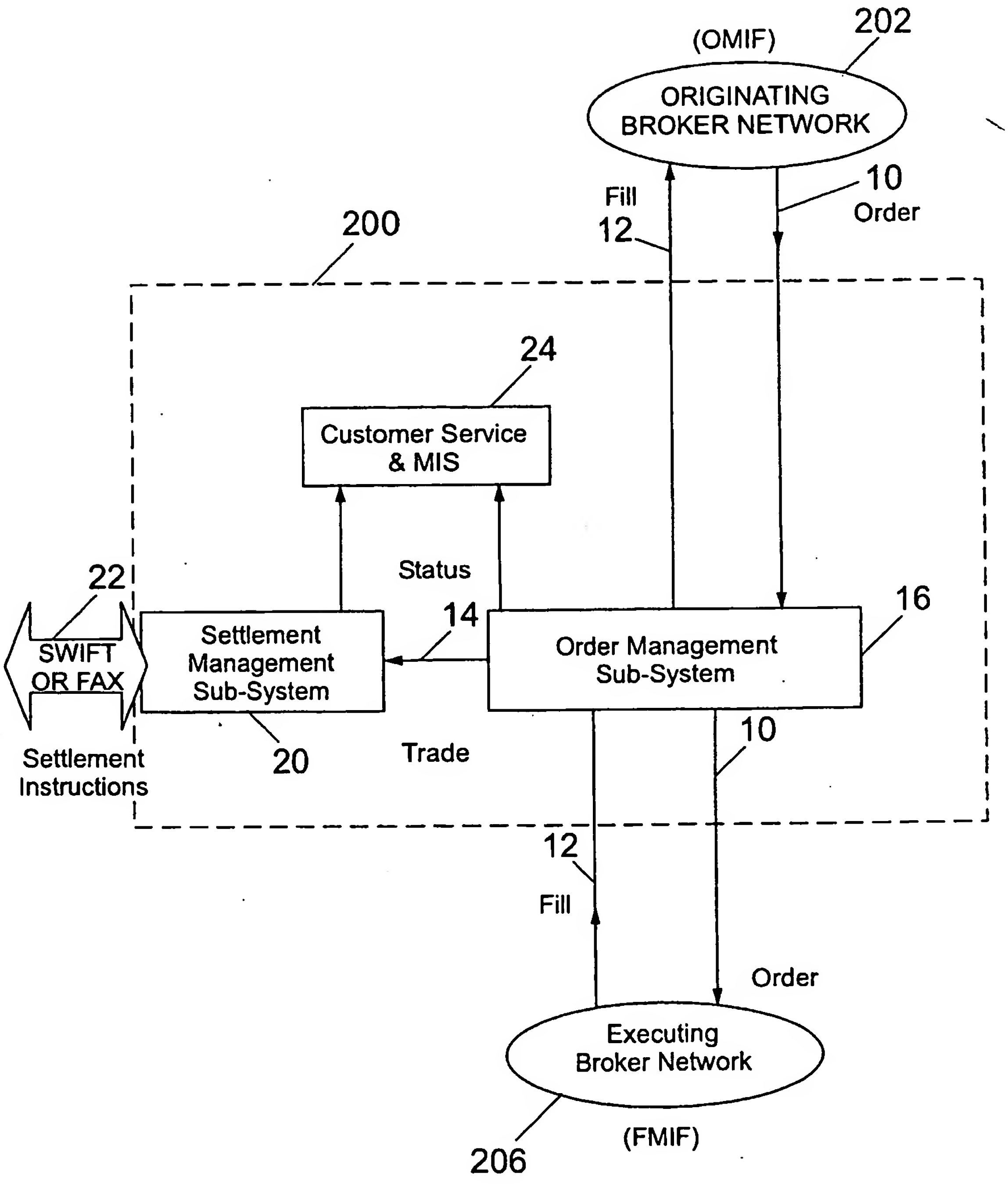
(OMIF)
202
ORIGINATING BROKER NETWORK
Fill
12
10
Order
200
24
Customer Service & MIS
Status
22
SWIFT OR FAX
Settlement Management Sub-System
14
Order Management Sub-System
16
Settlement Instructions
20
Trade
10
12
Fill
Order
Executing Broker Network
(FMIF)
206

Fig. 2

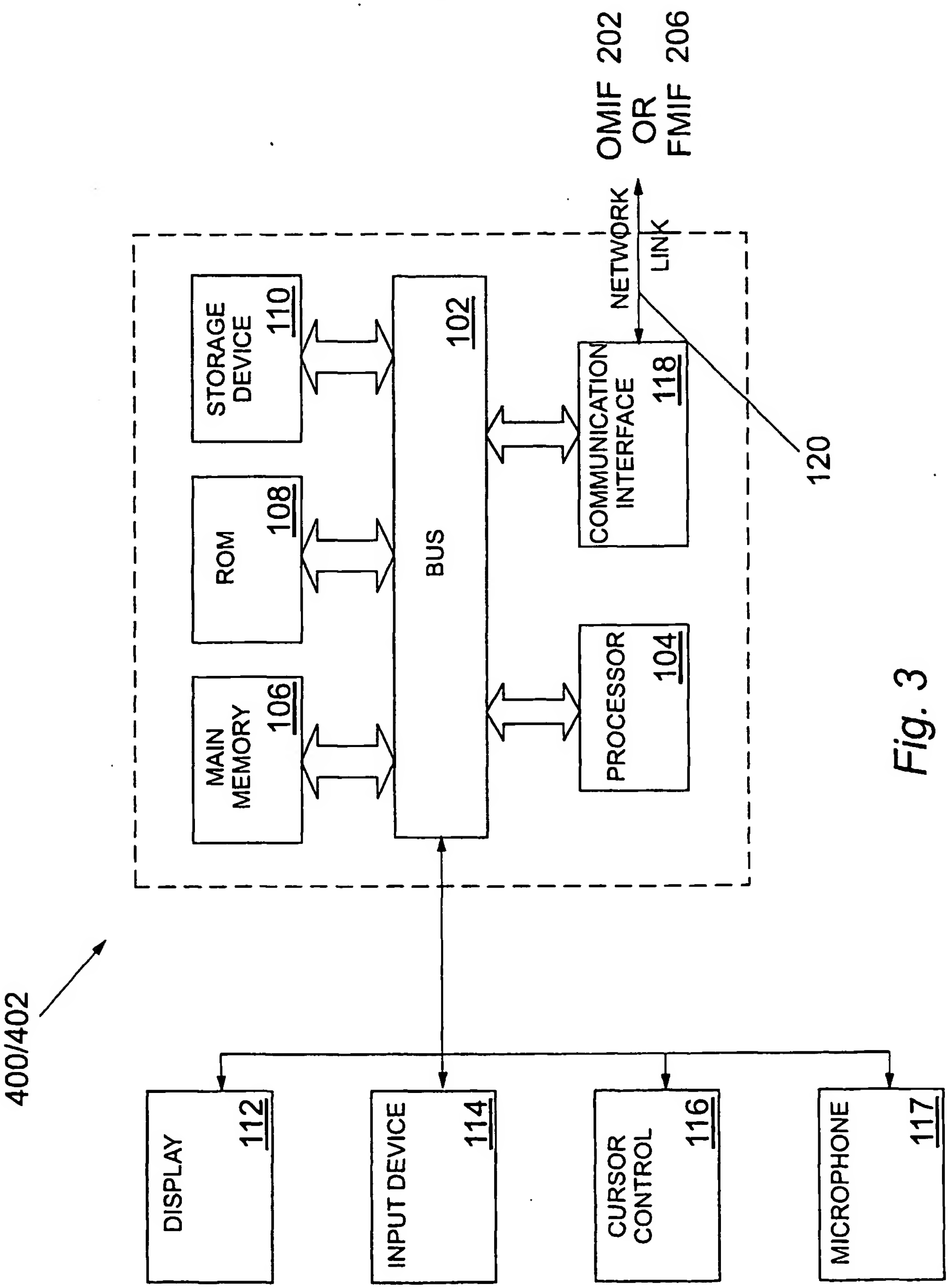
OMIF 202
OR
FMIF 206
NETWORK
LINK
STORAGE DEVICE
110
ROM
108
MAIN MEMORY
106
BUS
102
COMMUNICATION INTERFACE
118
120
PROCESSOR
104
400/402
DISPLAY
112
INPUT DEVICE
114
CURSOR CONTROL
116
MICROPHONE
117

Fig. 3

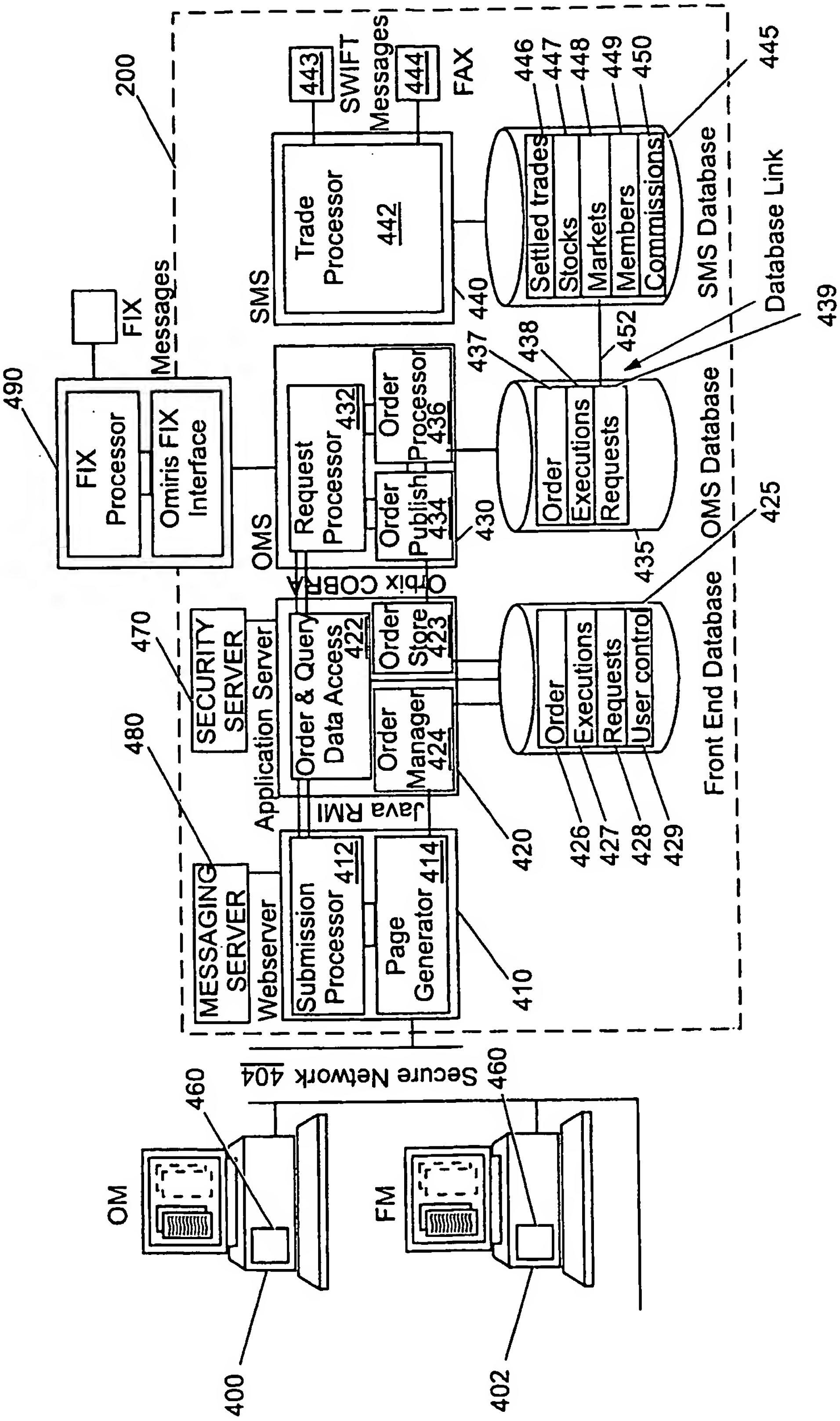
200
FIX Messages
490
FIX Processor
Omiris FIX Interface
SMS
Trade Processor
442
443
SWIFT Messages
444
FAX
446
447
448
449
450
445
Settled trades
Stocks
Markets
Members
Commissions
SMS Database
Database Link
440
452
439
437
438
OMS
Request Processor 432
Order Processor 436
Order Publish 434
430
Order
Executions
Requests
435
OMS Database
Orbix COBRA
470
SECURITY SERVER
Application Server
Order & Query Data Access 422
Order Store 423
Order Manager 424
420
Order
Executions
Requests
User control
425
Front End Database
426
427
428
429
480
MESSAGING SERVER
Webserver
Submission Processor 412
Page Generator 414
Java RMI
410
Secure Network 404
460
460
OM
FM
400
402

Fig. 4

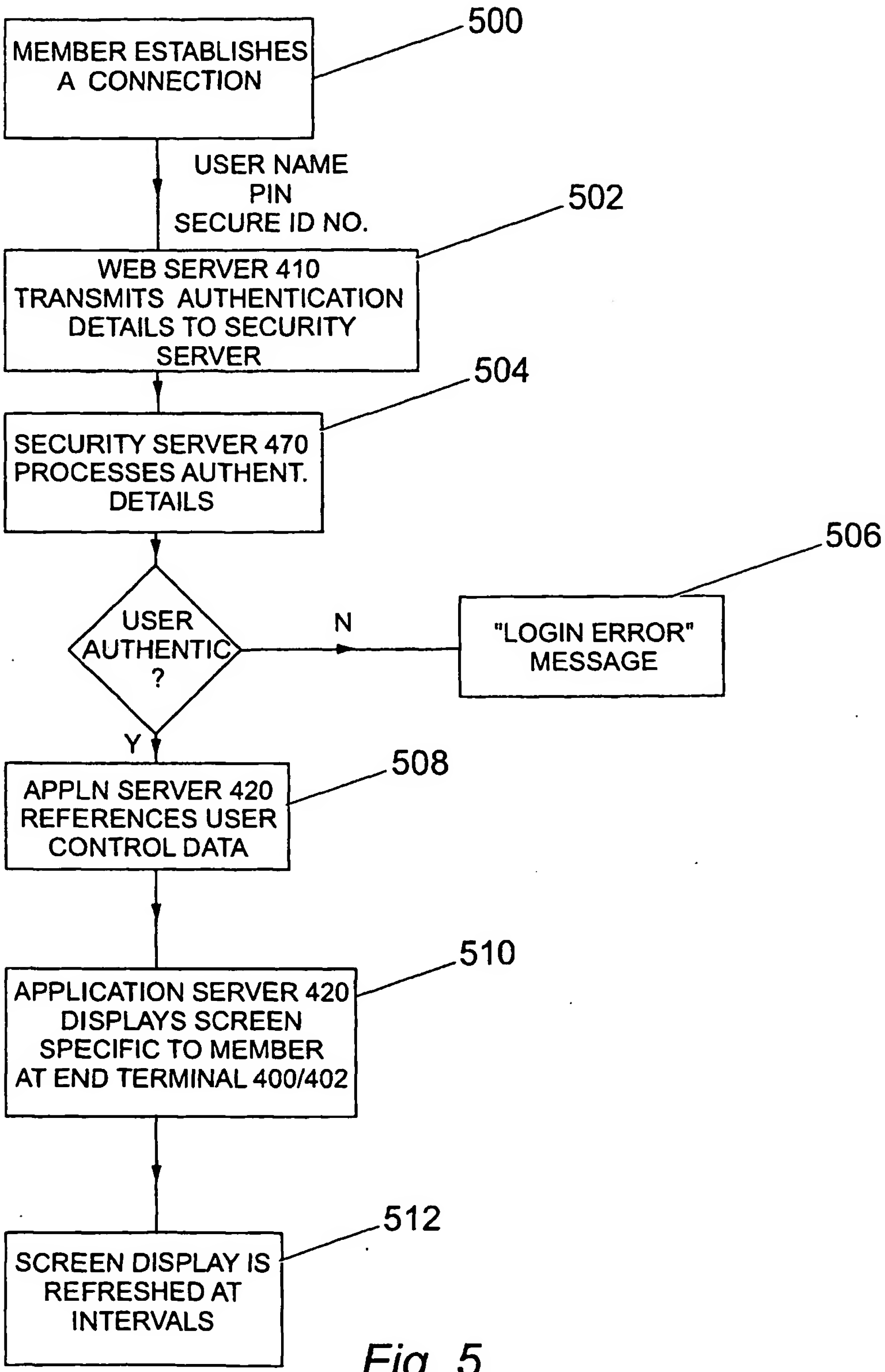
MEMBER ESTABLISHES A CONNECTION
500
USER NAME
PIN
SECURE ID NO.
WEB SERVER 410 TRANSMITS AUTHENTICATION DETAILS TO SECURITY SERVER
502
SECURITY SERVER 470 PROCESSES AUTHENT. DETAILS
504
USER AUTHENTIC ?
N
"LOGIN ERROR" MESSAGE
506
Y
APPLN SERVER 420 REFERENCES USER CONTROL DATA
508
APPLICATION SERVER 420 DISPLAYS SCREEN SPECIFIC TO MEMBER AT END TERMINAL 400/402
510
SCREEN DISPLAY IS REFRESHED AT INTERVALS
512

Fig. 5

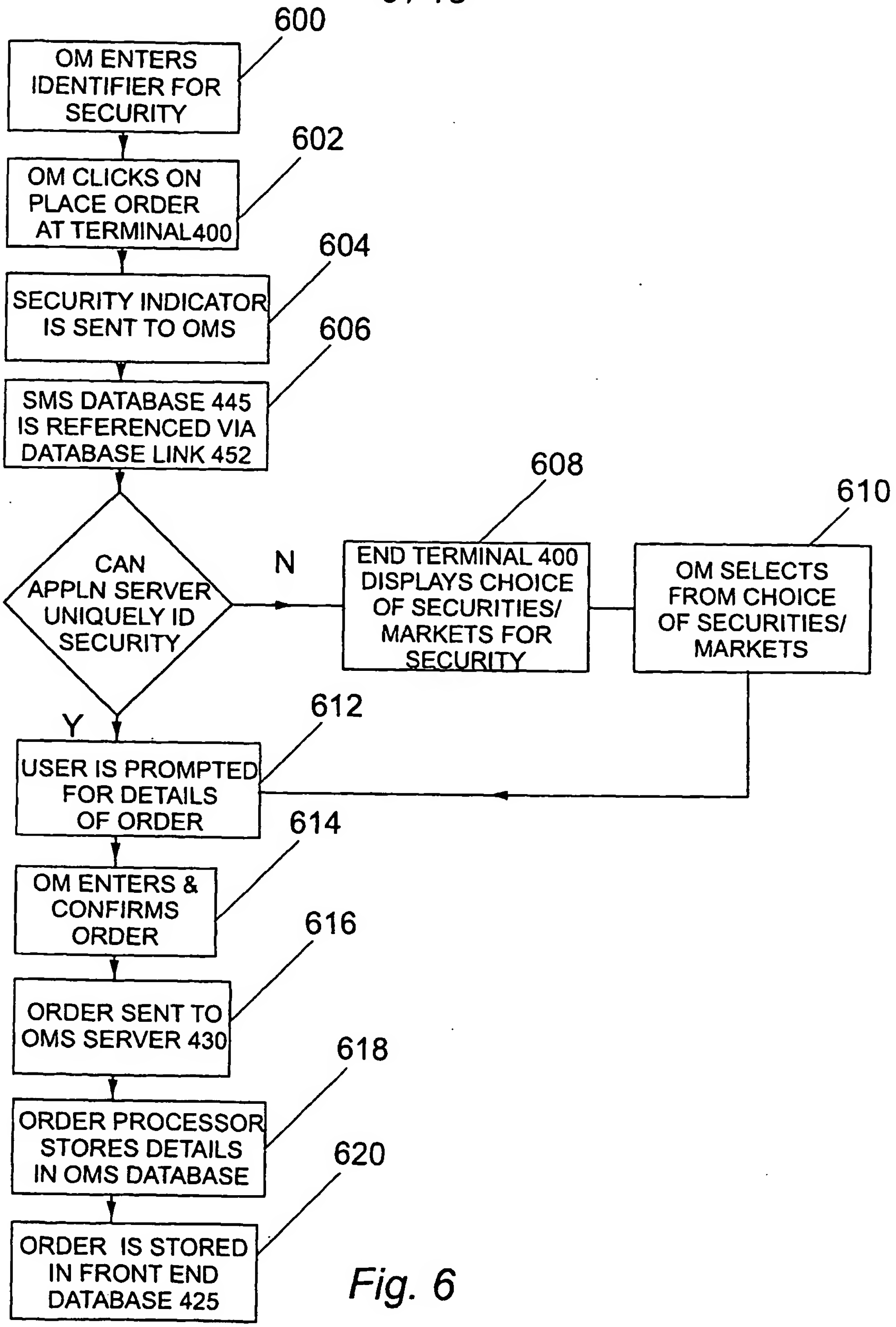
600
OM ENTERS IDENTIFIER FOR SECURITY
602
OM CLICKS ON PLACE ORDER AT TERMINAL400
604
SECURITY INDICATOR IS SENT TO OMS
606
SMS DATABASE 445 IS REFERENCED VIA DATABASE LINK 452
CAN APPLN SERVER UNIQUELY ID SECURITY
N
608
END TERMINAL 400 DISPLAYS CHOICE OF SECURITIES/ MARKETS FOR SECURITY
610
OM SELECTS FROM CHOICE OF SECURITIES/ MARKETS
Y
612
USER IS PROMPTED FOR DETAILS OF ORDER
614
OM ENTERS & CONFIRMS ORDER
616
ORDER SENT TO OMS SERVER 430
618
ORDER PROCESSOR STORES DETAILS IN OMS DATABASE
620
ORDER IS STORED IN FRONT END DATABASE 425

Fig. 6

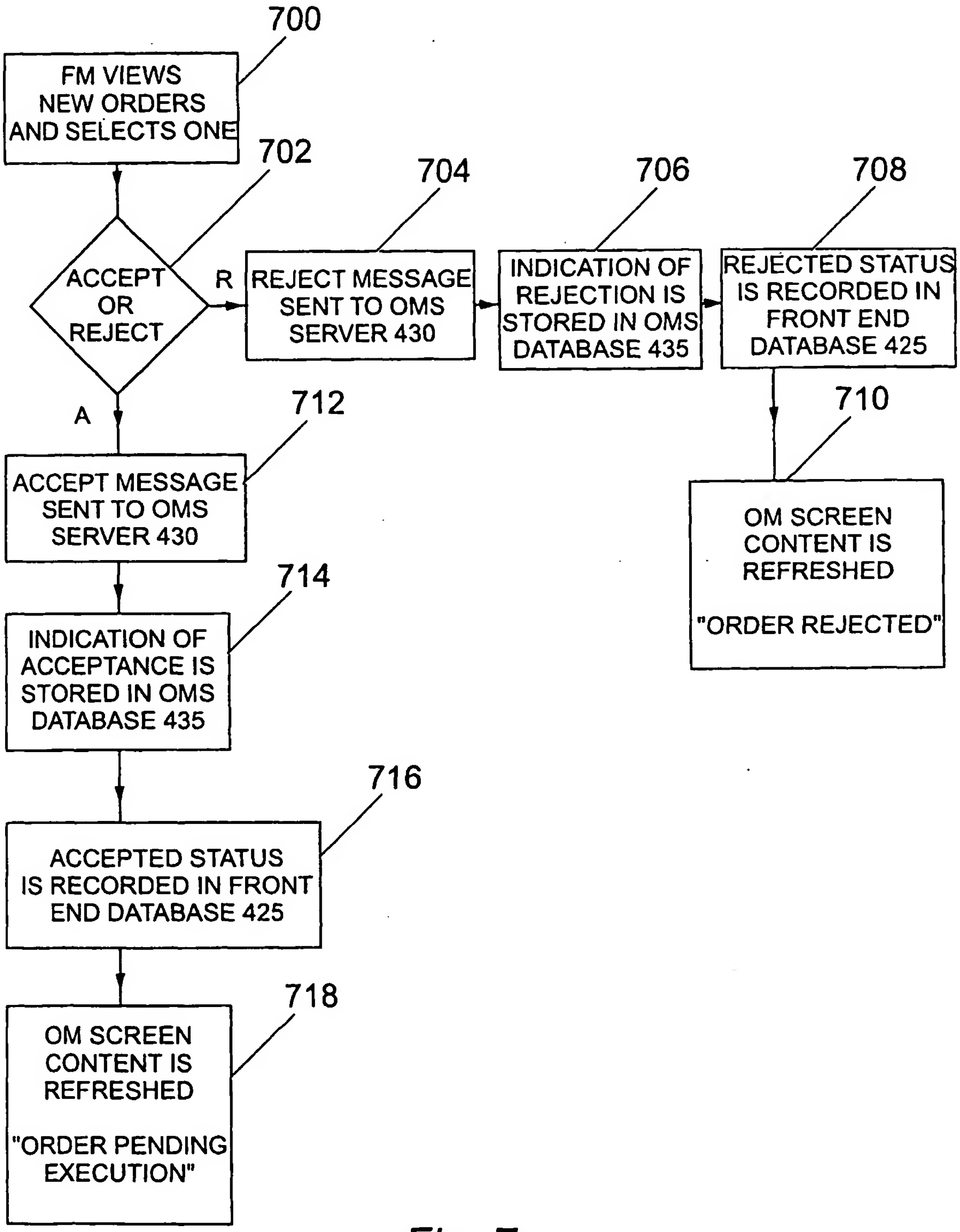
700
FM VIEWS NEW ORDERS AND SELECTS ONE
702
ACCEPT OR REJECT
R
704
REJECT MESSAGE SENT TO OMS SERVER 430
706
INDICATION OF REJECTION IS STORED IN OMS DATABASE 435
708
REJECTED STATUS IS RECORDED IN FRONT END DATABASE 425
710
OM SCREEN CONTENT IS REFRESHED "ORDER REJECTED"
A
712
ACCEPT MESSAGE SENT TO OMS SERVER 430
714
INDICATION OF ACCEPTANCE IS STORED IN OMS DATABASE 435
716
ACCEPTED STATUS IS RECORDED IN FRONT END DATABASE 425
718
OM SCREEN CONTENT IS REFRESHED "ORDER PENDING EXECUTION"

Fig. 7

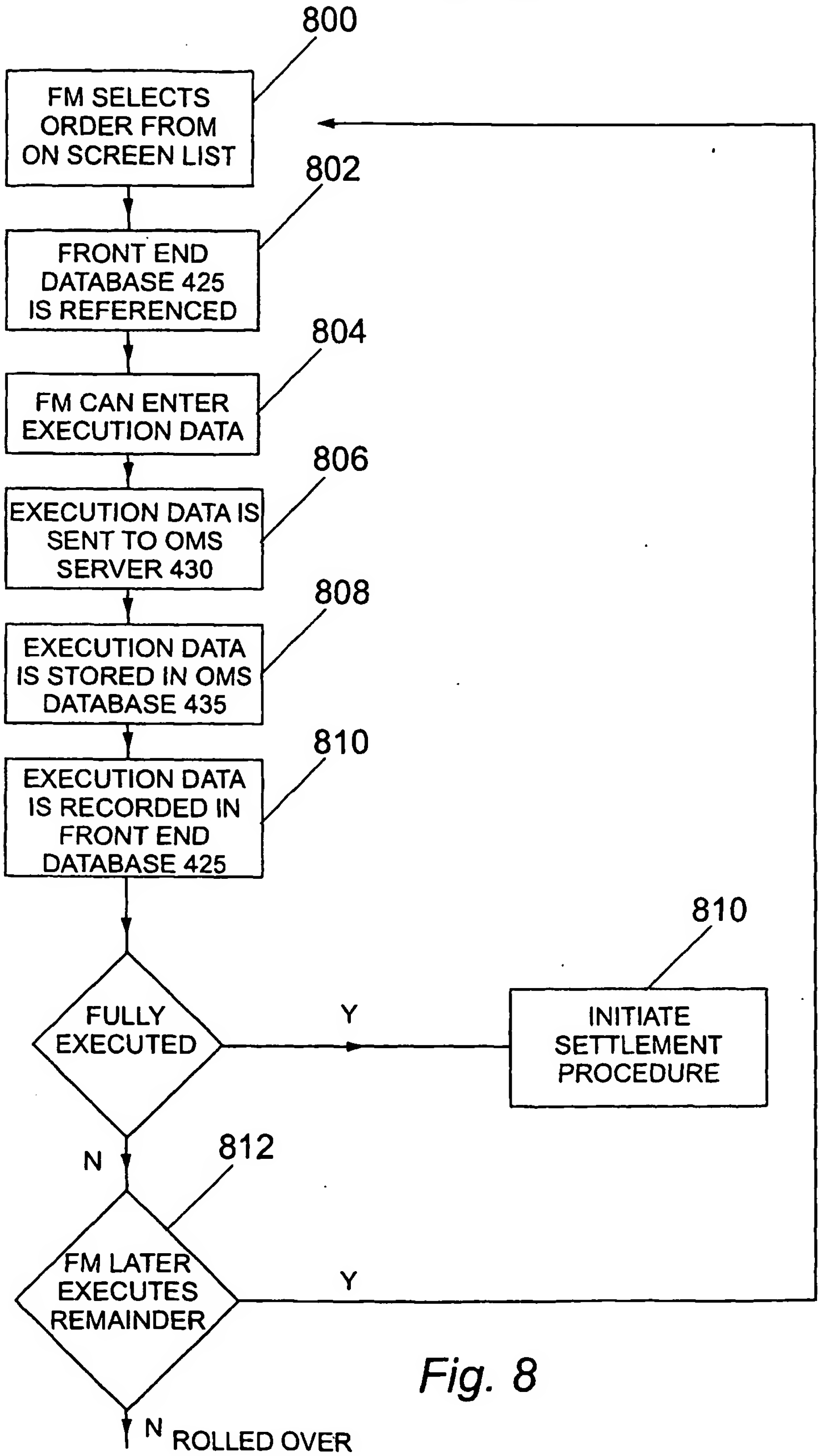
800
FM SELECTS ORDER FROM ON SCREEN LIST
802
FRONT END DATABASE 425 IS REFERENCED
804
FM CAN ENTER EXECUTION DATA
806
EXECUTION DATA IS SENT TO OMS SERVER 430
808
EXECUTION DATA IS STORED IN OMS DATABASE 435
810
EXECUTION DATA IS RECORDED IN FRONT END DATABASE 425
FULLY EXECUTED
Y
810
INITIATE SETTLEMENT PROCEDURE
N
812
FM LATER EXECUTES REMAINDER
Y
N
ROLLED OVER

Fig. 8

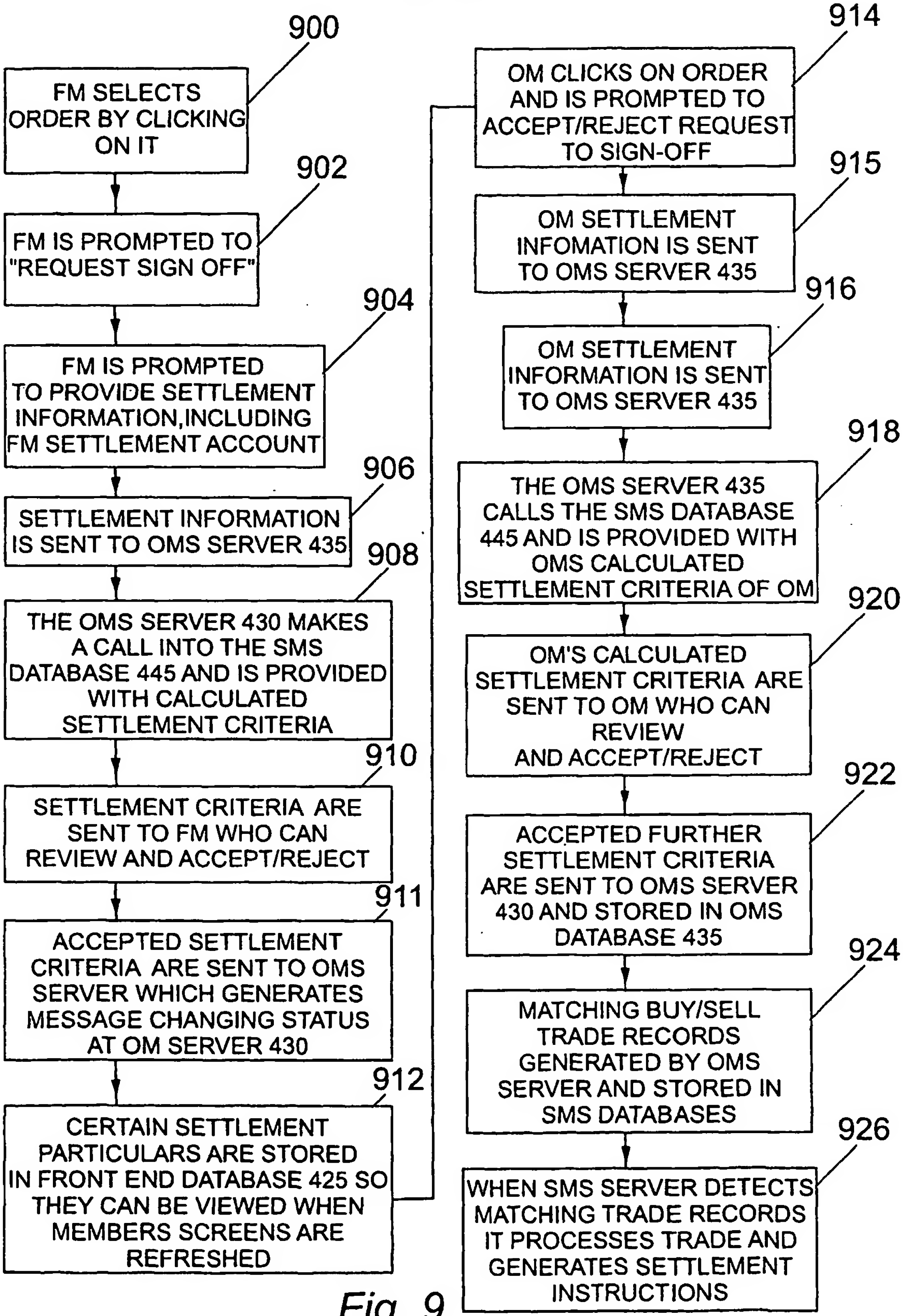
900
FM SELECTS ORDER BY CLICKING ON IT
902
FM IS PROMPTED TO "REQUEST SIGN OFF"
904
FM IS PROMPTED TO PROVIDE SETTLEMENT INFORMATION,INCLUDING FM SETTLEMENT ACCOUNT
906
SETTLEMENT INFORMATION IS SENT TO OMS SERVER 435
908
THE OMS SERVER 430 MAKES A CALL INTO THE SMS DATABASE 445 AND IS PROVIDED WITH CALCULATED SETTLEMENT CRITERIA
910
SETTLEMENT CRITERIA ARE SENT TO FM WHO CAN REVIEW AND ACCEPT/REJECT
911
ACCEPTED SETTLEMENT CRITERIA ARE SENT TO OMS SERVER WHICH GENERATES MESSAGE CHANGING STATUS AT OM SERVER 430
912
CERTAIN SETTLEMENT PARTICULARS ARE STORED IN FRONT END DATABASE 425 SO THEY CAN BE VIEWED WHEN MEMBERS SCREENS ARE REFRESHED
914
OM CLICKS ON ORDER AND IS PROMPTED TO ACCEPT/REJECT REQUEST TO SIGN-OFF
915
OM SETTLEMENT INFOMATION IS SENT TO OMS SERVER 435
916
OM SETTLEMENT INFORMATION IS SENT TO OMS SERVER 435
918
THE OMS SERVER 435 CALLS THE SMS DATABASE 445 AND IS PROVIDED WITH OMS CALCULATED SETTLEMENT CRITERIA OF OM
920
OM'S CALCULATED SETTLEMENT CRITERIA ARE SENT TO OM WHO CAN REVIEW AND ACCEPT/REJECT
922
ACCEPTED FURTHER SETTLEMENT CRITERIA ARE SENT TO OMS SERVER 430 AND STORED IN OMS DATABASE 435
924
MATCHING BUY/SELL TRADE RECORDS GENERATED BY OMS SERVER AND STORED IN SMS DATABASES
926
WHEN SMS SERVER DETECTS MATCHING TRADE RECORDS IT PROCESSES TRADE AND GENERATES SETTLEMENT INSTRUCTIONS

Fig. 9

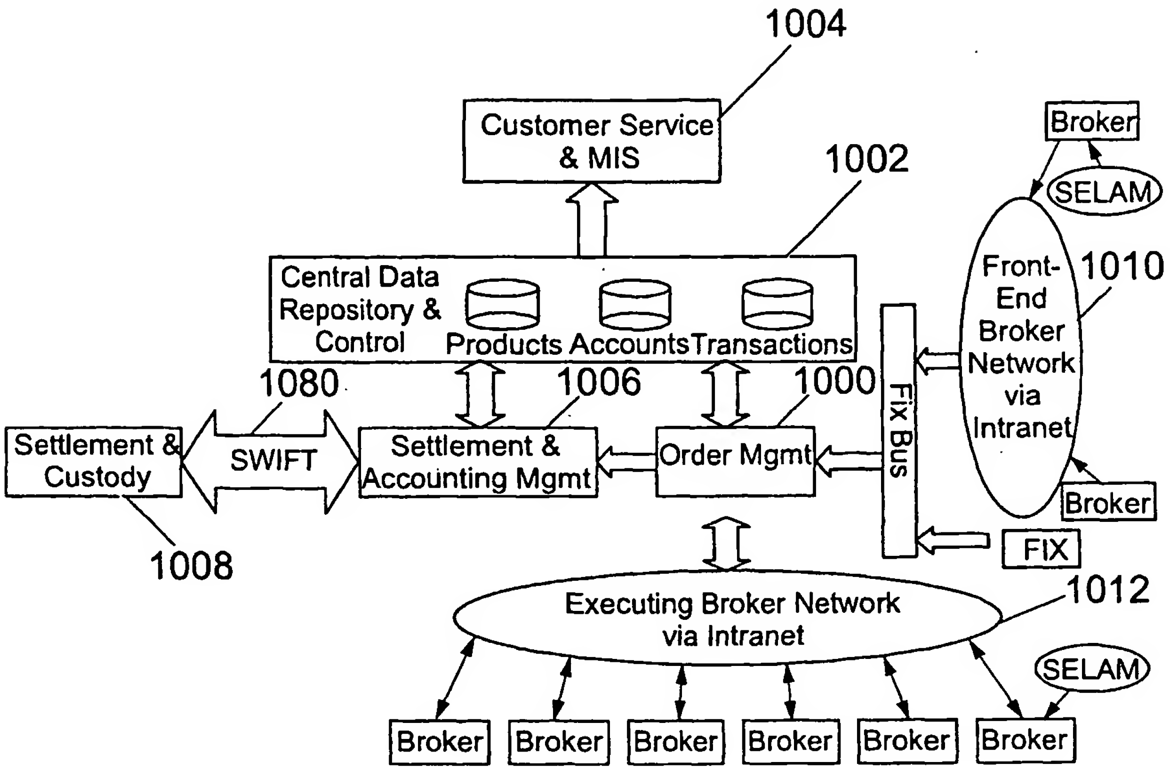
1004
Customer Service & MIS
1002
Broker
SELAM
Central Data Repository & Control
Products
Accounts
Transactions
Front-End Broker Network via Intranet
1010
1080
1006
1000
Fix Bus
Settlement & Custody
SWIFT
Settlement & Accounting Mgmt
Order Mgmt
Broker
1008
FIX
Executing Broker Network via Intranet
1012
SELAM
Broker
Broker
Broker
Broker
Broker
Broker

Fig. 10

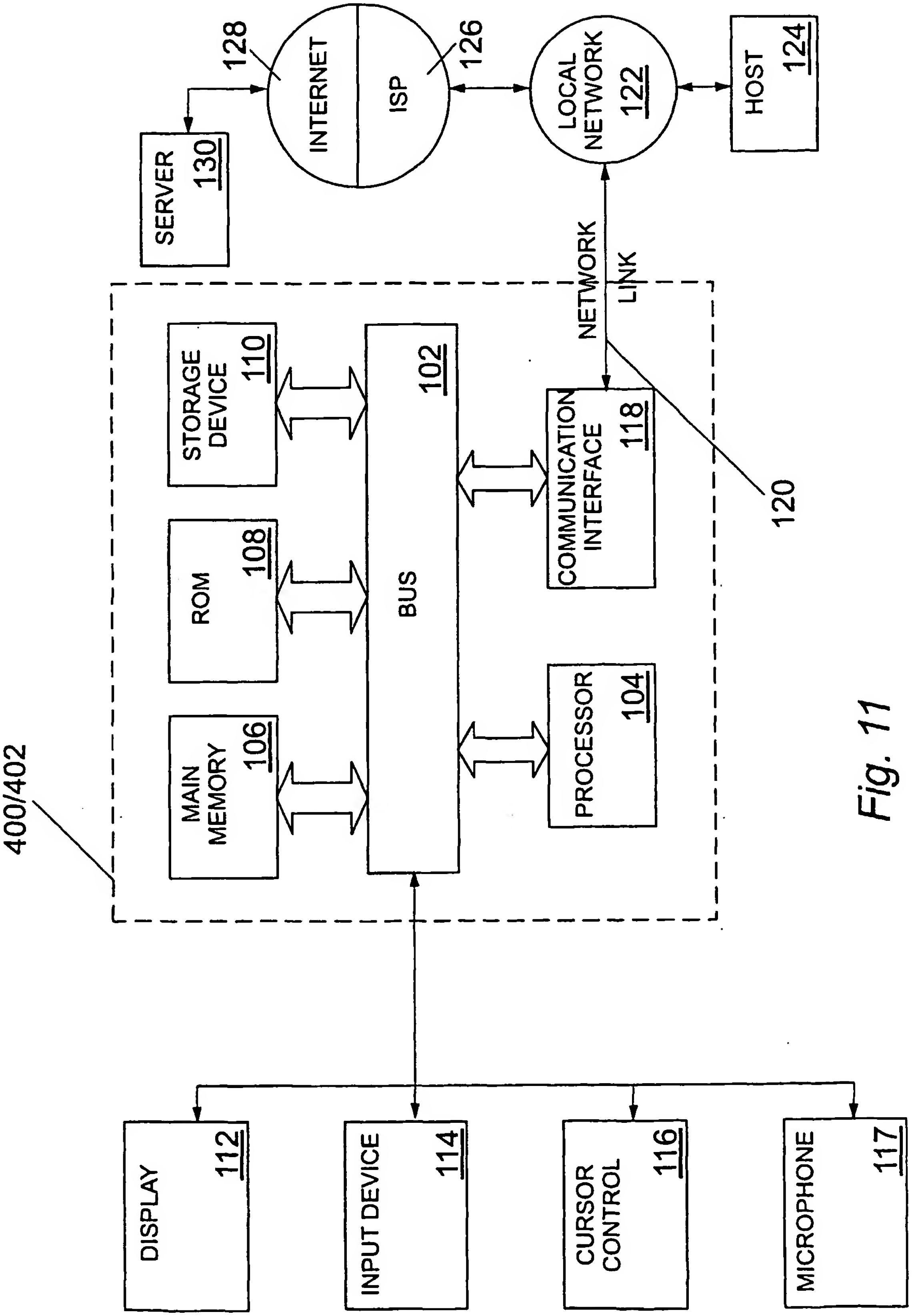
SERVER
130
128
INTERNET
ISP
126
LOCAL NETWORK
122
HOST
124
NETWORK LINK
120
400/402
STORAGE DEVICE
110
ROM
108
MAIN MEMORY
106
BUS
102
COMMUNICATION INTERFACE
118
PROCESSOR
104
DISPLAY
112
INPUT DEVICE
114
CURSOR CONTROL
116
MICROPHONE
117

Fig. 11

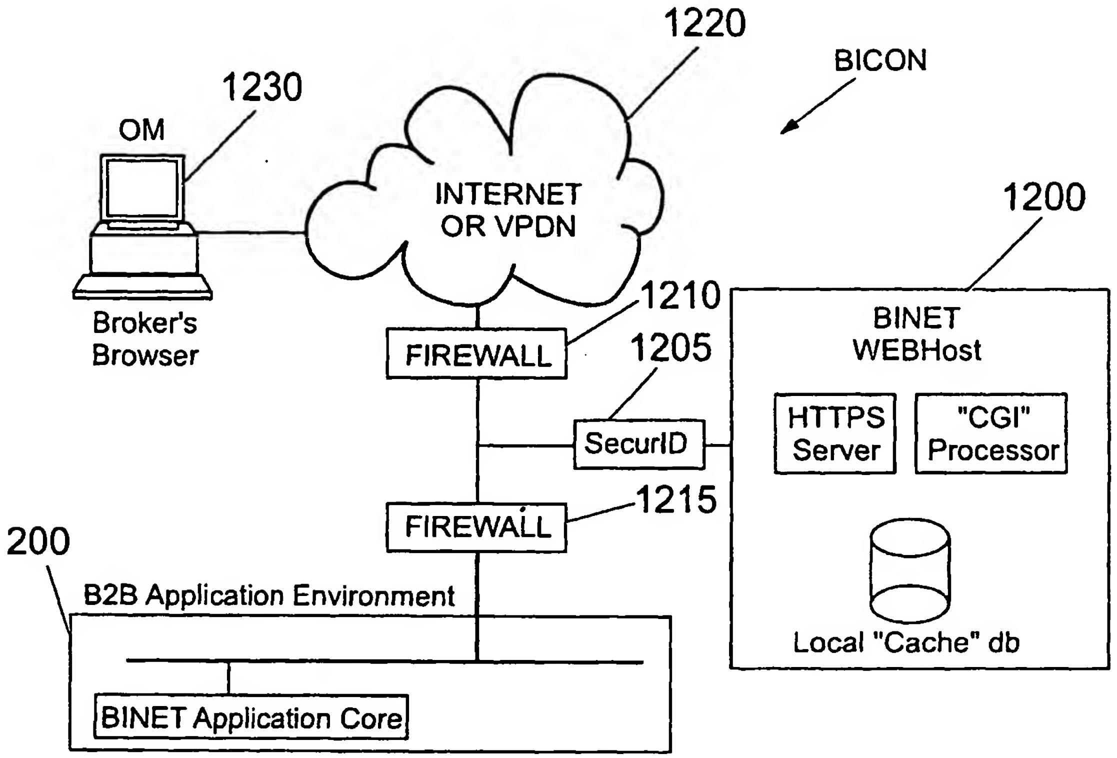
1220
BICON
1230
OM
INTERNET OR VPDN
1200
1210
Broker's Browser
BINET WEBHost
FIREWALL
1205
HTTPS Server
"CGI" Processor
SecurID
1215
FIREWALL
200
B2B Application Environment
Local "Cache" db
BINET Application Core

Fig. 12

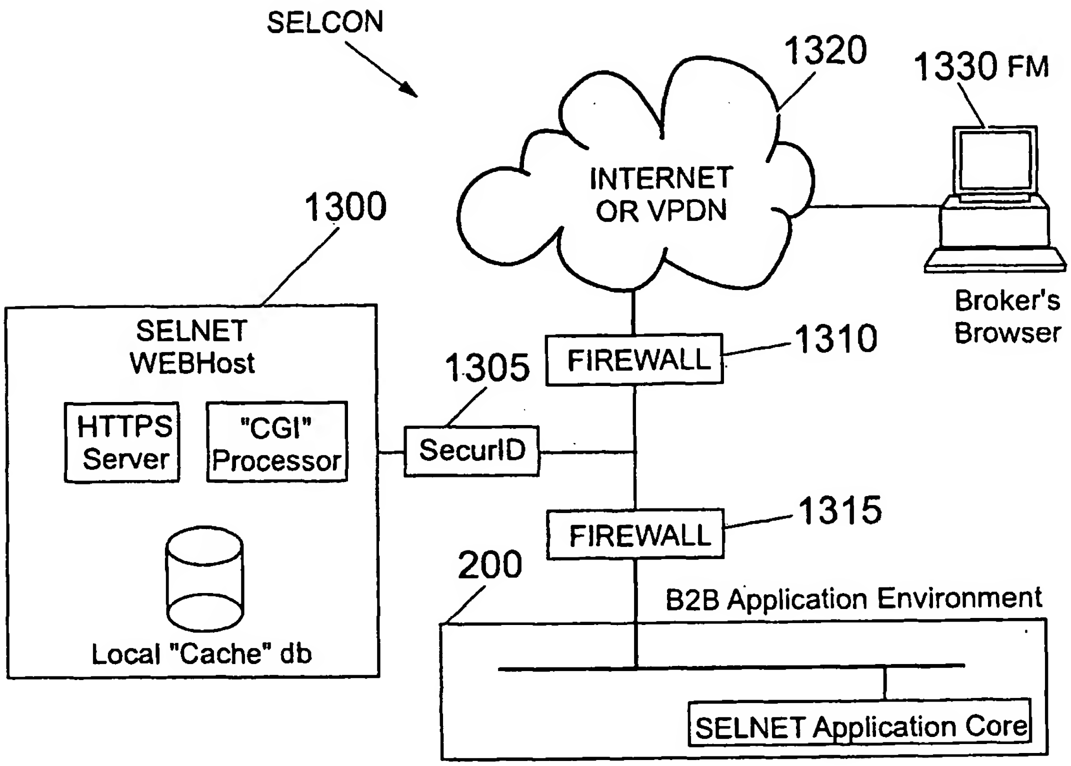
SELCON
1320
1330 FM
INTERNET
OR VPDN
1300
Broker's
Browser
SELNET
WEBHost
FIREWALL
1310
1305
HTTPS
Server
"CGI"
Processor
SecurID
FIREWALL
1315
200
B2B Application Environment
Local "Cache" db
SELNET Application Core

Fig. 13